Sensitivity-based algorithm for modelling of plants with uncertainty and for control system design

Golesorkhi, Nader

Sensitivity-Based Algorithm For Modelling Of Plants With Uncertainty And For Control System Design

A Thesis Submitted For The

Degree Of Doctor Of Philosophy

In The Faculty Of Engineering, University Of London

By

Nader Golesorkhi

June 1993

Department Of Electronics And Electrical Engineering

King's College London, University Of London

London WC2R 2LS

Abstract

The use of a sensitivity criterion in the design of control systems has been proposed for many years. Various researchers have incorporated sensitivity criteria in their systems' designs; such as constant gain feed-back controller and pole-assignment, which are mostly system specific. Until now, however, no application-independent sensitivity-based design method has been established.

This thesis conducts an in-depth study of a novel computer-aided 3-d graphical method of analysing plants with uncertainty in the complex frequency domain. Central to this method has been the development of a novel, highly efficient root locus family algorithm based on sensitivity formulations. The algorithm has been successfully implemented for modelling plants undergoing any combination of simultaneous variations of loop gain, and positions of open-loop poles and/or of open-loop zeros. By interpreting plant uncertainties in terms of fuzzy zones surrounding nominal positions of the plant open-loop poles and zeros, the designer/analyst, can view a succession of either interactive or automated 3-d graphical displays which reveal step by step account of the plant's behaviour.

A comprehensive range of both deterministic and stochastic plants have been investigated primarily for evaluating the algorithm. The result of these studies however, have been organised in a form of plant knowledge-base sensitivity classification (KBSC) which may be used as an interactive CAD tool for assessing different controller configurations for optimum system robustness.

Furthermore, it has been demonstrated that the algorithm can effectively model plants undergoing a general class of fault arising from gradual deterioration of characteristics.

The work presented highlights the algorithm's potential as a CAD tool for fault-tolerant systems. On the other hand, the KBSC lays the foundation for an automatic identification of the optimum pole/zero pattern, leading to a new generation of robust adaptive control schemes for real-life plants.

Acknowledgements

This thesis is dedicated to my wife Mitra, whose love and encouragement have given enormous boost to my research effort. I would like to thanks her for her love and patience and for her valuable help in the preparation of this thesis.

Special thanks are due to Dr. F. B. Khalafalla- my supervisor at the department of Electronics and Electrical Engineering, Kings College, University of London- for his guidance, enthusiasm, and never ending support throughout this thesis.

I would like to express my gratitude to members of staff for their guidance and the assistance with the facilities:: Dr. J. Fenly, M. Jassop and S. Layson of the computer centre and the technicians of the department of Electronics and Electrical Engineering.

I wish to thank my parents and parents-in-law for their support and encouragement during my higher education.

Finally, I thank my friends Angela, Jeffrey and Lucy for their support and patience, and those of my colleagues and friends, in particular Hamid Dabiry, who made my study at Kings interesting.

Table Of Contents

List Of Figures

List Of Tables

Chapter 1

Introduction

1.1 General

The insatiable appetite for highly sophisticated products, accompanied with the sharply increased awareness of the consequential issues involving health, safety and environment, as detected in the recent years, have inspired a fundamental review of contemporary design priorities and techniques for both plants and products. It is no longer sufficient to aim for optimum performance, robustness and cost effectiveness. Reliability, adaptability and fault tolerance are now perceived as key objectives particularly for new intelligent applications. Therefore, there is an increasing demand for more sophisticated CAD tools capable of handling a widely diverse and complex requirements and yet easy to use. These call for new advances in system modelling techniques and in man-computer dialogues, on one hand, and for establishing an integrated approach to the design with in-built performance evaluation and quality assurance measures, on the other. Among the most recent research advances [1] to [19], the role of the root-sensitivity (or eigenvalue sensitivity), in particular, as a design criterion for systems with plant parameter variations, has received considerable attention although it is a less direct measure of system performance. This is mainly because of the widespread use of pole-placement or eigenvalue assignment design-based techniques which give the designer the ability to specify system stability and general characteristics of the time-domain response. The investigators [6] to [19] have unanimously reported that the incorporation of a sensitivity criterion has enhanced the characteristics of their particular designs through modal insensitivity. Although the concepts of sensitivity in design have been known for many years, however, to the best

of our knowledge, no attempt so far has been made to incorporate the sensitivity within a generalised design/analysis framework suitable for diverse control system applications in real-life environment.

The main objective of this thesis is, therefore, an attempt to address this deficiency. Central to the research work is the development of a novel, application-independent, sensitivity-based algorithm capable of modelling plants with uncertainty - Golesorkhi and Khalafallah [3, 4] - and/or undergoing some common modes of faults - Golesorkhi and Khalafallah [5], of automatic design decisions in terms of composite performance-sensitivity and of analysis and design evaluations, as will unfold throughout the thesis. By incorporating highly automated or interactive screen graphic displays of 2-d and 3-d, together with a host of default criteria, specification-loci and design strategies, the algorithm has evolved into a highly flexible, cost-effective CAD system as will be demonstrated in the last three chapters. As a direct consequence, a graphic knowledge-base sensitivity classification (KBSC) of single-loop feedback systems, having plant configurations of increasing order with widely diverse pole/zero patterns and delays, has been generated (see Appendix B). This KBSC can be easily integrated with the developed CAD system to provide an additional design tool. In demonstrating the effectiveness of the CAD system - in modelling both deterministic and stochastic plants, in simulation and analysis of fault scenarios, in design of robust controllers of fixed and adaptive configurations for plants with uncertainties, delays and/or undergoing fault conditions widely diverse in nature, rate and degree of severity, as supported by the extensive studies and applications carried out in the last three chapters - provide adequate evidence that the primary goals set out for this research work have been successfully achieved. Finally to be able to produce, almost in real time, the optimum decision path for the adaptive controller, concurrently with the operation path of a feasible fault-tolerant control system for a given plant fault path (see chapter 7), highlights the potential of the developed algorithm for a possible

future adaptation to operate as on-line intelligent controller to diverse real-life applications whilst meeting health, safety and environmental standards.

1.2 System Models And Uncertainties

A practical control system based on a fixed theoretical design, when implemented, may generally encounter variety of problems such as non-minimum-phase plants, i.e., the existence of zeros in the right half of complex frequency domain, so the inverse is unstable. There are inevitably unmodelled dynamics that produce substantial uncertainty, usually at high frequency; and that sensor noise and input signal level constraints limit the achievable benefits of feed-back. Thus to propose a system that can reduce uncertainties; possible causes of uncertainties must be identified. Consider a real physical system that may be represented by either of the following:

a) Ideal physical model obtained by schematically decomposing the real system into ideal building blocks; composed of components such as resistors, masses, beams, electrons, and so on.

b) Ideal mathematical model obtained by applying natural laws to the ideal physical model; typically composed of non-linear partial differential equations.

c) Reduced mathematical model obtained from the ideal mathematical model by linearisation, lumping, and so on; usually a rational transfer function.

No mathematical system can precisely model a real physical system, there is always uncertainty. Thus the exact output of a real physical system cannot be predicted even if the input is known. Uncertainty arises from two sources: unknown or unpredictable inputs such as disturbance, noise, parametric uncertainty representing lack of precise

knowledge of the actual system parameters and unpredictable dynamics representing high frequency uncertainties.

The parametric changes may be incremental or large, for example small parameter variations are found in mechanical systems with fluctuating values of friction, hydraulic system with properties of fluid varying with temperature and loading. In electro-mechanical devices such as an ultrasonic transducer, the change in the characteristics of the device is normally due to the fluctuating values of the acoustic impedance that changes with the medium characteristics according to the environmental conditions. In some cases even small changes in the value of the plant parameters may cause considerable changes in the system behaviour.

If the control system design is based on a fixed set of assumed model parameters, then perturbations due from these parameters can cause deterioration in the system's performance, or even result in an unstable system. A generalised block diagram of a single-loop feedback system subject to disturbance and uncertainties is illustrated in figure 1.1

A model should predict the input-output response in such a way that a design based on the model will represent the real physical system closely; minimising modelling errors. This requires the plant model to be non-deterministic, having uncertainty built in explicitly. To minimise the effect of parameter uncertainty or perturbations, sensitivity constraints should be incorporated in the design.

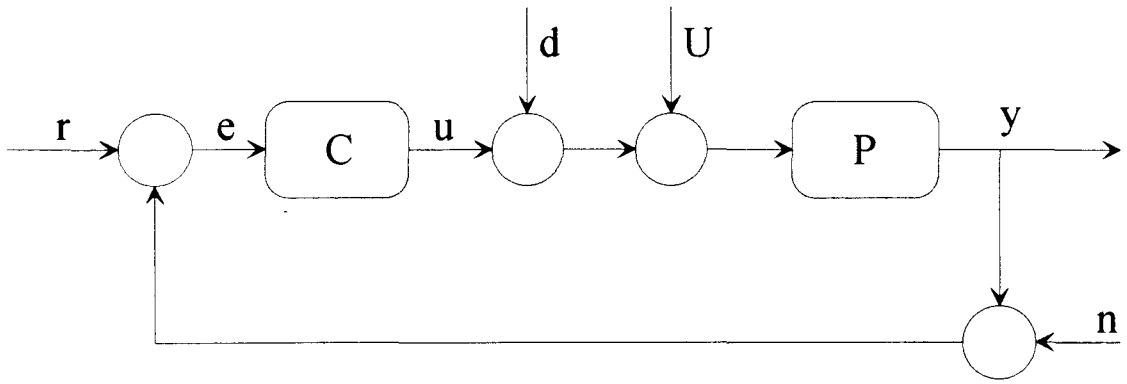


Figure 1.1 Single-loop feedback system subject to uncertainty

where

r	reference input	e	tracking error
C	controller	u	control signal, controller output
d	plant disturbance	U	uncertainties
P	plant	y	plant output
n	sensor noise		

1.3 Sensitivity: Definition And Classification

Sensitivity provides a better insight into the significant relationships between system parameters, perturbations of their values and their subsequent effects on the system performance. Sensitivity analysis allows a priori assessment to be made of the effect of any system modification on the system performance.

The classification of sensitivity is given by Gourishankar and Ramar [1]. There are three types of sensitivities: performance index sensitivity, trajectory sensitivity and eigenvalue sensitivity. A drawback in using performance index sensitivity is those important variations in system behaviour caused by plant parameter changes may not

be reflected in the performance index - Rynaski [2]. Trajectory sensitivity is a measure of variation in the response of a system due to plant parameter variations. Eigenvalue sensitivity or root sensitivity is the closed-loop poles sensitivity of a system to variation in plant parameters. Although eigenvalue sensitivity is a less direct measure of system performance, it is being recognised to be a very useful measure of system performance. This approach has received considerable attention that several investigators have used this concept to relate parameter changes to the characteristic roots of the system.

1.4 Overview Of Background On Sensitivity Approach To The Design

Recently a large number of researchers have employed some sensitivity criterion to achieve specific design targets. Gourishankar and Ramar [1], for instance, incorporated sensitivity in the design of a controller to assign poles of a closed-loop system in the complex frequency plane and through examples showed the superiority of a system with sensitivity design compared to the design where sensitivity aspects were not considered. Morgan [6], on the other hand, developed relevant formulations for sensitivity analysis of time-invariant multivariable by a system of first-order differential equations and sensitivity coefficients for the transfer function matrix in terms of the system parameters. Reddy [7] explains, yet, how the sensitivity functions can be used to determine the transient response of a linear dynamic, time-invariant, multivariable feed-back controller for selected closed-loop eigenvalues. Owens and O'Reilly [8] discuss the problem of closed-loop system eigenvalue sensitivity to small parameter variations and propose a constant gain feedback matrix that assigns the closed-loop's eigenvalue to desired locations with low sensitivity to variations in some, but not all the elements of the open-loop state-space model. Elaborating on the same theme, Shah, Fisher and Seborg [9] proposed the design of a constant gain multivariable feedback controller that makes selected closed-loop eigenvalues invariant to unknown

perturbation of arbitrary magnitude in system parameters. Further, Qui and Gourishank [10] developed a method for designing constant-gain feedback controllers for linear systems that assign the closed-loop poles to specified locations with minimum eigenvalues sensitivities to plant parameter variations. Verde and Maritelo [11] followed by presenting closed-loop eigenvalue assignment with minimum trajectory sensitivity and through an example concluded that "a system without sensitivity and robustness considerations become unstable for a 50% variation in an independent parameter vector". Wellstead and Daley [12] advanced the design of a pole placement controller by transforming to a state space system, performing an output feedback design and transforming the controller back to polynomial form. They found that by examining the closed-loop transient response of the system, the robust controller design has performed significantly better than the standard controller. Duan [13] expanded the robust pole-assignment technique with minimum sensitivity for linear systems through output feedback where the inclusion of closed-loop eigenvalues in the optimisation parameters have significantly improved the robustness of the pole-assignment problem. Kautsky, Nichols and Dooren [14] work describes a method of determining robust solution to the problem of multi-input state-feedback pole-assignment. The solution reduces the sensitivity of assigned poles to perturbations in the system and gain matrices and provides improvement in the transient response and stability margin. Howze and Carin [15] considered the problems of eigenvalue assignment achieving modal insensitivity, i.e., the response mode shapes are insensitive to small variations in plant parameter. Cavin and Bhattacharyya [16], on the other hand, produced an algorithm for the problem of pole-assignment that makes the eigenvector well conditioned with respect to inversion, equivalently and maximally orthonormal. This method is hoped to be a step in the direction of transforming pole assignment into a design tool for control systems. Crossley and porter [17] produced further derivations for eigenvalue and eigenvector sensitivity coefficients for the eigen problem case associated with linear systems. While Chen, Wang and Lu [18] proposed the design of a perfect model-matching control system with minimal

sensitivity to external disturbances. Gahinet, Laub, Kenney and Hewer [19] advanced a method for determining a sensitivity measure of stable discrete-time Lyapunov eigen problem of the open-loop state matrix.

To summarise, the majority of the above investigators have incorporated some sensitivity criterion in their design in an attempt to minimise perturbations in their particular control system design. The most common approach seems to be based on the techniques of pole placement or eigenvalue assignment that have been widely used in both single variable and multivariable design applications because they give the designer the ability to specify system stability and general characteristics of the time-domain response.

It is noted, however, that in all the above reported literature, the work has been geared to a specific system and no general design method incorporating a sensitivity criterion, uncertainty and perturbations seems to emerge. This deficiency has provided both the motivation and the aims for the work presented in this thesis as will be defined in the subsequent sections.

1.5 Chaos And Catastrophe Theory

All real systems are non-linear to start with and are therefore chaotic. The non-linear phenomena includes the multiple attractors observable in a single system; chaotic long-term behaviour and its underlying order and structure; and discontinuous jump and hysteresis phenomena.

Time evolutions of dynamical systems must normally be modelled by non-linear equations for which closed-form analytical solutions are unobtainable. They are however readily integrated numerically by routine computer algorithms, so that the

response from given starting conditions is easily established. But the starting conditions of a real system are never known precisely, and may be totally unknown. Hence, what can happen in the evolution of a system, and how this is influenced by the starting conditions are very important considerations. The major guidance from dynamical systems theory is the concept of an attractor. Typical, dissipitive dynamical systems exhibit a start-up transient, after which the motion settles down towards some form of long-term recurrent behaviour. Motions from adjacent starts tend to converge towards stable attracting solutions.

On the other hand, a dynamic system can have an instability of a special kind. As a result of this instability, various dynamic characteristics of the system vary in time randomly. Chaos thus arises as an internal property of the system. Many systems are in a slowly evolving environment, so their coefficients and parameters undergo gradual change. Then if the evolving system is in a steady state of equilibrium, periodic oscillation or chaos, the prediction of any sudden change is of crucial importance. Such bifurcation of behaviour will occur when the phase portrait undergoes a qualitative change of topological form at a point of structural instability. An important distinction between those catastrophic bifurcations at which there is a finite rapid dynamic jump to a new steady state, and subtle bifurcation in which the change in response manifests itself in the smooth growth of a new local attractor after the bifurcation point.

Catastrophe theory provides a universal method for the study of all jump transitions, dis-continuities, and sudden qualitative changes. Catastrophe theory has been applied to such diverse fields as, for instance, biological, ecological, social and economic sciences as well as traditional fields of mechanics, physics, and chemistry.

The origin of Catastrophe theory lie in Whitney's theory of singularities of smooth mappings and Poincare' and Andronov's theory of bifurcations of dynamical systems Arnold[78,79].

Singularity theory is a far-reaching extension of the study of functions at maximum and minimum points. In Whitney's theory functions are replaced by mappings, i.e., collections of several functions of several variables.

The word bifurcation is used in a broad sense for designating all possible qualitative reorganisations or metamorphoses of various objects resulting from changing the parameters on which they depend. Catastrophies, however, are violent sudden changes representing discontinuous responses of systems to smooth changes in the external conditions.

Whitney's Singularity Theory

A mapping of a surface onto a plane associates to each point of the surface a point of the plane. If the point on the surface is given by co-ordinates (x_1, x_2) and the point on the plane by (y_1, y_2), then the mapping is given by a pair of functions $y_1=f_1(x_1, x_2)$ and $y_2=f_2(x_1, x_2)$. The mapping is said to be smooth if these functions are smooth (i.e. differentiable a sufficient number of times, such as polynomials).

Generally, only two singularities are encountered. All other singularities disintegrate under small movements of the body or of the angle of the projection while these two types are stable and persist after small deformations of the mapping.

The first singularity called Whitney fold arising at equatorial point when a sphere is projected onto a plane. The second, Whitney singularity, the cusp, arises at non-

equatorial points for example, when projecting a surface that has parts with different-sized curvaceous, onto a plane.

Whitney proved that the cusp is stable, i.e., every nearby mapping has a similar singularity at the appropriate point (singularity such that, in suitable co-ordinates, a deformation/mapping in a neighbourhood of the appropriate point is described by the same formulae as those describing the original mapping.

Whitney also proved that every singularity of a smooth mapping of a surface onto a plane, after an appropriate small perturbation, splits into folds and cusps.

Bifurcations Of Equilibrium States

A point of phase space is called a state of the system. A vector field at this point indicates the speed of change of the state.

Equilibrium states are points that the vector is zero and they do not change with time. Some equilibrium states are unstable. The oscillation represented by a closed curve is called a limit cycle.

Curves in phase space representing sequential states are called phase curves. In a neighbourhood of a point that is not an equilibrium state, the partition of phase space into phase curves is just like a partition into parallel lines. The family of phase curves can be transformed into a family of parallel lines by a smooth change of co-ordinates. In a neighbourhood of an equilibrium point Poincare' has shown that all the more complex patterns disintegrate into combinations of the ones after a small generic perturbation such as: focus (stable), source (unstable), saddle and sink.

Systems describing real evolutionary processes are as a rule generic. In fact such systems always depend on parameters that are never known exactly. A small generic change of parameters transforms a system that is not generic into one that is. For systems depending on one or more parameters, we consider the space of all systems divided into domains of generic systems. The dividing surfaces represent degenerate systems; under a small change of parameter values a degenerate system becomes non-degenerate.

Thus although for each individual value of the parameter a system can be transformed by a small perturbation into a non-degenerate one, this cannot be done simultaneously for all values: every curve close to the one considered intersects the dividing boundary at a close value of the parameter (any degeneracy is removable by a small perturbation at the given value of the parameter, but appears again at some nearby value).

Hence, when a whole family rather than an individual system is considered, the degeneracy's are not removable. If the family is a one parameter family then the unremovable degeneracies are only the simplest being represented by boundaries of co-dimension one (i.e., given by one equation) in the space of all systems. One can save one parameter families from more complex degenerate systems by small perturbations.

Poincare's general considerations are applicable not only to the study of equilibrium states of evolutionary systems but to a large part of mathematical analysis in general. First we study the generic case, then degeneracies of co-dimension one, then two, etc. In this study of degenerate systems we must not restrict our study to the moment of degeneracy but must include a description of the metamorphosis that takes place as the parameter passes through the critical value.

It is noted that if stable equilibrium states describe the steady state of real systems (say in economics, ecology or chemistry) then combination with an unstable equilibrium

state must cause the system to jump to a different state. When the parameter is changed the equilibrium states in the neighbourhood considered suddenly disappear. Loss of stability of an equilibrium state on change of parameter is not necessarily associated with the bifurcation of this state. An equilibrium state can lose stability without even interacting with another state.

Singularities Of Accessibility Boundaries

A control system in phase space may be defined by a set of vectors at every point of the space called the indicatrix of permissible velocities. In this context, Arnold[78] discussed how to reach a target in the shortest possible time. In general case it may not be possible to reach the target under certain initial conditions. The points of the phase space from which the target can be reached is called accessibility domain.

The boundary of the accessibility domain can have singularities even when neither the target nor the indicatrix field have singularities. A classification given below by A. A. Davidov[80] of the singularities of accessibility boundaries in generic controlled systems on phase planes when the indicatrix and the target are smooth curves .

Of the four types of boundary singularities, three are given by the following simple formulae for appropriate choices of local co-ordinates on the plane:

1) $y = |x|$

2) $y = x|x|$

3) $y = x^2|x|$

The fourth type of singularity contains an arbitrary smooth function B in two variables x, y and three parameter a, b and $\alpha > 1$.

4)

$$a(x-\sqrt{y})^{\alpha}=x/\alpha+y\beta-\sqrt{y},\quad x\geq 0;$$

$$b(|x|-\sqrt{y})^{\alpha}=x/\alpha+y\beta+\sqrt{y},\quad x<0.$$

Arnold [78] has also shown the error of R. Thom's statement that "In nature one meets only stable phenomena and therefore in every problem one should study the stable cases, rejecting the others as being unrealisable". In the above case the first three singularities are stable whereas the fourth is not. Nevertheless all four types are encountered equally often and the study of the last no less important than of the others.

1.6 Research Contributions

The main objective of research work presented in this thesis is to develop an application-independent sensitivity-based CAD system for the analysis and design of robust, fault-tolerant, adaptive control systems to operate in real environment. A summary of the original contributions and general advances is listed below:

* The theoretical development of a new sensitivity-based algorithm capable of modelling plants with uncertainty and of simulation of common modes of faults efficiently.
* Implementation and integration of the new algorithm into a highly versatile CAD system, incorporating interactive/automatic 2-d and 3-d composite graphics, automated decision-making procedures, self-evaluation and a fairly comprehensive system analysis.

* Production of plant knowledge-base sensitivity classification (KBSC) which may be used as an interactive CAD tool for assessing different controller configurations for optimum system robustness and/or fault tolerance.

* Use of the new algorithm to- develop a coherent approach to modelling and analysis of feedback systems subject to parameter perturbations and/or fault modes operating in a noisy environment.

* Development and application of generalised strategies for computer aided design of fixed-configuration or adaptive controllers with an in-built evaluation of composite performance/quality assurance for fault-tolerant stochastic plants operating in real environment.

Chapter 2

Theoretical Development Of A Sensitivity-Based Algorithm For Performance Prediction Of Feed-Back Systems

2.1 Introduction

In chapter 1, we saw the incorporation of some sensitivity criterion in various control system design applications. This chapter, details the derivations of a set of sensitivity related formulations which emphasise the relationships between the residues at the closed-loop poles, and their respective sensitivities, and how this information could be used to predict the new locations of the closed- loop poles when one or more of the parameters had undergone a relatively small change. The formulations relate specifically to three distinct parameter variations: in loop gain, in open-loop poles and in open-loop zeros which rely on the following definitions of sensitivity:

Sensitivity of the ith closed-loop pole q_i due to variation in loop gain $\mathbf{S}_{K}^{qi} = \frac{\delta q_i}{\delta K/K}$

Sensitivity of the ith C.L. pole q_i due to changes in open-loop pole P_j $\mathbf{S}_{P_j}^{q_i} = \frac{\delta q_i}{\delta P_j}$

Sensitivity of the ith C.L. pole q_i due to changes in open-loop zero Z_j $\mathbf{S}_{Z_j}^{q_i} = \frac{\delta q_i}{\delta Z_j}$

The results of these formulations were then utilised in developing a sensitivity based algorithm that incorporates the above three types of variations to produce a novel estimation method for concurrent generation of root-loci and root-sensitivity loci which yield efficiently fast graphic displays in response to any single or to any combination of simultaneous variations.

The steps involved in transforming the algorithm into a form suitable for computer implementation are detailed, ready for integration into a new CAD tool suitable for design of robust, fault-tolerant control systems having plants subject to all types of uncertainty and/or undergoing a diverse range of faults belonging to a common class, and operate in a real noisy environment, as will be discussed in chapters 6 & 7.

2.1.1 Preliminaries

The overall transfer function of a control system represented by a block diagram or by a signal-flow graph model can be obtained directly using Mason's loop rule [55], given as follows:

$$T_{C.L.}(S) = \frac{C(S)}{R(S)} = \frac{\left(\sum_k T_k(S)\Delta_k(S)\right)}{\Delta(S)}$$

where

$T_k(S) \underset{=}{\Delta}$ transfer function of the kth forward path from the input to the output; a forward path cannot contain any feedback loop.

$\Delta(S) \underset{=}{\Delta}$ is the determinant of the system

= 1-sum of all individual loop transfer functions + sum of the products of the transfer functions of all possible sets of two non-touching loops - sum of the products of the transfer functions of all possible sets of three non-touching loops + ···

$\Delta_k(S)$ = all terms in $\Delta(S)$ that do not have elements or path common with an element or path in $T_k(S)$.

where the closed-loop poles lie at the solution of $\Delta(S) = 0$.

2.2 Single-Input Single-Output Feedback Systems

For a SISO feedback system reduced to a single loop as shown in figure 2.1, the closed-loop transfer function will be $T_{C.L.}(S) = \dfrac{G(S)}{1 + G(S)\,H(S)}$

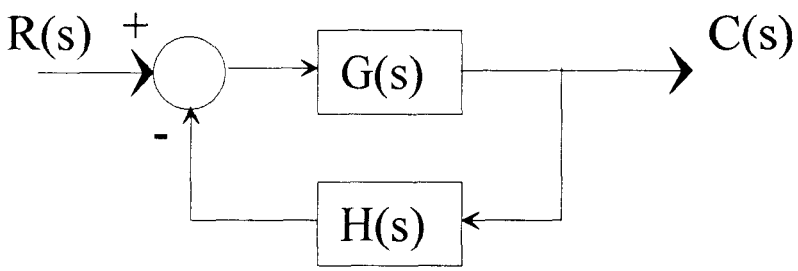


Figure 2.1 A SISO Of A Single-Loop System With Feedback H(S)

Giving the characteristic equation:

$$\Delta(S) = 1 + G(S)\,H(S) \qquad (1)$$

which may be re-written in a factorised general form as:

$$\Delta(S) = 1 + \frac{\left(K(S-Z_1)\ldots(S-Z_r)^a\ldots(S-Z_m)\ldots\{(S-Z_{C1})(S-Z_{C1}^*)\} \right.}{\left. \{(S-Z_{Cr})(S-Z_{Cr}^*)\}^A \ldots\ldots \{(S-Z_{CM})(S-Z_{CM}^*)\}\, e^{-T_dS} \right)} \qquad (2)$$
$$\frac{\left((S-P_1)\ldots(S-P_r)^b\ldots(S-P_n)\ldots\ldots\{(S-P_{C1})(S-P_{C1}^*)\} \right.}{\left. \{(S-P_{Cr})(S-P_{Cr}^*)\}^B \ldots\ldots \{(S-P_{CN})(S-P_{CN}^*)\} \right)}$$

Where

$Z_1 \ldots Z_m$: Are distinct (single) real, open-loop (O.L.) zeros.

Z_r^a : Are multiple real open-loop zeros, wherein a is the order of multiplicity.

$\{ Z_{C1}, Z_{C1}^* \} \ldots \{ Z_{CM}, Z_{CM}^* \}$: Are distinct complex conjugate pairs of open-loop zeros.

$\{ Z_{Cr}, Z_{Cr}^* \}^A$: Are multiple of complex conjugate pairs of open-loop zeros, wherein A is the order of multiplicity.

T_d : Is a fixed pure delay in the forward path, in seconds.

$P_1 \ldots P_n$: Are distinct real open-loop poles.

P_r^b : Are multiple real open-loop poles, wherein b is the order of multiplicity.

$\{ P_{C1}, P_{C1}^* \} \ldots \{ P_{CN}, P_{CN}^* \}$: Are distinct complex conjugate pairs of open-loop poles.

$\{P_{Cr}, P_{Cr}^*) \}^B$: Are multiple of complex conjugate pairs of open-loop poles, wherein B is the order of multiplicity.

2.3 Reformulation Of The Characteristic Equation In Terms Of The Residues Of The Closed-Loop Poles

By considering multiple poles (or zeros) as a cluster of distinct poles (or zeros) regularly distributed in the vicinity of the spot, each displaced by an infinitesimal distance ε approaching zero, it is possible to reduce the poles and zeros to only two types: **distinct real or distinct complex conjugate pairs**. Using this description, the characteristic equation (1) can now be simplified to the following standard format:

$$\Delta(S) = 1 + \frac{\left(K(S-Z_1)\ldots(S-Z_m)\ldots\{(S-Z_{C1})(S-Z_{C1}^*)\} \{(S-Z_{CM})(S-Z_{CM}^*)\} \right) e^{-T_d S}}{\left((S-P_1)\ldots\ldots(S-P_n)\ldots\ldots\{(S-P_{C1})(S-P_{C1}^*)\} \{(S-P_{CN})(S-P_{CN}^*)\} \right)} \qquad (3)$$

This may be re-written in terms of the residues at the distinct poles, thus

$$\Delta(S) = 1+K\left(\frac{R_1}{(S-P_1)}+\cdots\frac{Rn}{(S-P_n)}+\cdots+\left(\frac{R_{C1}}{(S-P_{C1})}+\frac{R_{C1}^*}{(S-P_{C1}^*)}\right)+\cdots+\left(\frac{R_{CN}}{(S-P_{CN})}+\frac{R_{CN}^*}{(S-P_{CN}^*)}\right)\right) \quad \ldots.(4)$$

where

R_n : Is a real residue at the distinct real pole P_n.

R_{CN} : Is a complex residue in general at the distinct complex conjugate pole P_{CN}.

R_{CN}^*: Is a complex residue (conjugate to R_{CN})at the distinct complex pole P_{CN}^* (which is conjugate to P_{CN})

Since the characteristic equation of the closed-loop system is **unique to the system structure; independent of the output point**, it is possible to arrive at a generalised set of formulations by focusing on unity-feedback systems (see figure 2.2). Therefore, the closed-loop transfer function ($T_{C.L.}$) can now be written as:

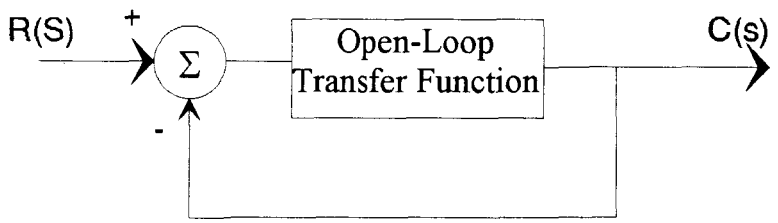


Figure 2.2 A Unity Feed-Back System

$$T_{C.L.}(S) = \frac{G(S)H(S)}{1+G(S)H(S)} \quad (5)$$

where G(S)H(S) id defined as the open-loop transfer function of the system. For both G(S) and H(S) to be physically realisable, each should be possible to be expressed as rational polynomials. Therefore, the product G(S)H(S) should also be possible to be expanded as a ratio of (numerator open-loop) / (denominator open-loop). Equation (5) can, therefore, be expressed as:

$$T_{C.L.}(S) = \frac{K(\text{numerator open loop}) / (\text{denominator open loop})}{1+K(\text{numerator open loop}) / (\text{denominator open loop})} \quad (6)$$

which can be reduced to:

$$T_{C.L.}(S)=\frac{K(\text{numerator open loop})}{(\text{denominator open loop})+K(\text{numerator open loop})} \qquad (7)$$

Through factorisation, equation (7) results in:

$$T_{C.L.}(S)=\frac{K[(S-Z_1)\cdots(S-Z_m)\cdots\{(S-Z_{C1})(S-Z_{C1}^{*})\}\cdots\{(S-Z_{CM})(S-Z_{CM}^{*})\}]e^{-T_dS}}{\{(S-P1)\cdots(S-Pn)\cdots\{(S-P_{C1})(S-P_{C1}^{*})\}\cdots\{(S-P_{CN})(S-P_{CN}^{*})\}+K[(S-Z_1).....(S-Z_m)...\{(S-Z_{C1})(S-Z_{C1}^{*})\}\cdots\{(S-Z_{CM})(S-Z_{CM}^{*})\}]e^{-T_dS}} \qquad \ldots.(8)$$

Expanding the denominator of $T_{C.L.}$ (S) into a polynomial then factorising the result, yield:

$$T_{C.L.}(S)=\frac{K[(S-Z_1).....(S-Z_m)...\{(S-Z_{C1})(S-Z_{C1}^{*})\}\cdots\{(S-Z_{CM})(S-Z_{CM}^{*})\}]e^{-T_dS}}{\{(S-q1)\cdots(S-qn)\cdots\{(S-q_{C1})(S-q_{C1}^{*})\}\cdots\{(S-q_{CN})(S-q_{CN}^{*})\}} \qquad \ldots.(9)$$

Where $q_1 \ldots q_n$: Are distinct real closed-loop poles.

$\{ q_{C1}, q_{C1}^{*}\}.....\{ q_{CM}, q_{CM}^{*} \}$: Are distinct complex conjugate pairs of closed-loop poles.

For particular values of K multiple real closed-loop poles or multiple complex conjugate pairs of closed-loop poles may result at what is referred to as real saddle (breakaway) points or complex conjugate pairs of saddle (breakaway) points.

These, however, can be approximated by a cluster of distinct closed-loop poles in the vicinity of these breakaway points, as explained earlier, without loss of generality.

Using this approach, however, will help avoid computation overflow errors since the values of the relevant residue as the closed-loop pole gets closer to the saddle point will be approaching infinity (as can be deduced from equation (9)).

It is also noted that for unity-feedback systems, the numerator of the closed-loop transfer function is the same as the numerator of the open-loop transfer function, thus:

$$T_{C.L.}(S)=\frac{K(\text{numerator open loop})}{(\text{Denominator closed loop})} \tag{10}$$

2.4 Sensitivity Of The Closed-Loop Poles Due To Variations Of The Loop-Gain K As A Function Of The Residues

The sensitivity of the ith closed-loop pole q_i due to variations of the loop-gain K has been already defined in section 2.1, which is reproduced here for convenience as follows:

$$S_K^{q_i}=\frac{\delta q_i}{\delta K/K}=K\frac{\delta q_i}{\delta K} \tag{11}$$

From equations (7) & (10) we obtain

$$\text{Den. open-loop} + K\,\text{Num. open-loop} = \text{Den. closed-loop} \tag{12}$$

Partial differentiation with respect to K results into:

$$0 + \text{Num. open-loop} = \frac{\delta}{\delta K}[\text{Den. closed-loop}] \tag{13}$$

But

$$\frac{\delta}{\delta K}\text{Ln}[\text{Den. closed-loop}] = \frac{\delta}{\delta K}\text{Ln}(S-q_1)+.....\frac{\delta}{\delta K}\text{Ln}(S-q_n)+.....$$
$$+\left(\frac{\delta}{\delta K}\text{Ln}(S-q_{C1})+\frac{\delta}{\delta K}\text{Ln}(S-q_{C1}^{*})\right)+...+\left(\frac{\delta}{\delta K}\text{Ln}(S-q_{CN})+\frac{\delta}{\delta K}\text{Ln}(S-q_{CN}^{*})\right) \tag{14}$$

Or

$$\frac{1}{\text{Den. closed loop}}\,x\,\frac{\delta[\text{Den. closed loop}]}{\delta K}=\frac{1}{(S-q_1)}\,x\,\frac{\delta(S-q_1)}{\delta K}+.....$$
$$+\frac{1}{(S-q_n)}\,x\,\frac{\delta(S-q_n)}{\delta K}+....+\left(\frac{1}{(S-q_{C1})}x\frac{\delta(S-q_{C1})}{\delta K}+\frac{1}{(S-q_{C1}^{*})}x\frac{\delta(S-q_{C1}^{*})}{\delta K}\right)+....$$
$$+\left(\frac{1}{(S-q_{CN})}x\frac{\delta(S-q_{CN})}{\delta K}+\frac{1}{(S-q_{CN}^{*})}x\frac{\delta(S-q_{CN}^{*})}{\delta K}\right) \tag{15}$$

where the terms of the right-hand side of the equation can be reduced to:

$$\text{R.H.S. of the equation (15)} = \frac{-1}{(S-q_1)} x \frac{\delta q_1}{\delta K} + \ldots\ldots + \frac{-1}{(S-q_n)} x \frac{\delta q_n}{\delta K} + \ldots\ldots$$
$$+\left(\frac{-1}{(S-q_{C1})} x \frac{\delta q_{C1}}{\delta K} + \frac{-1}{(S-q_{C1}^{*})} x \frac{\delta q_{C1}^{*}}{\delta K} \right) + \ldots\ldots$$
$$+\left(\frac{-1}{(S-q_{CN})} x \frac{\delta q_{CN}}{\delta K} + \frac{-1}{(S-q_{CN}^{*})} x \frac{\delta q_{CN^*}}{\delta K} \right) \qquad (16)$$

Multiply equation (15) throughout by K and using equations (11), (13) and (16), will give:

$$T_{C.L.}(S) = \frac{-1}{(S-q_1)} \cdot \mathbf{S}_{K}^{q_1} + \ldots\ldots + \frac{-1}{(S-q_n)} \cdot \mathbf{S}_{K}^{q_n} + \ldots\ldots$$
$$+\left(\frac{-1}{(S-q_{C1})} x \mathbf{S}_{K}^{q_{C1}} + \frac{-1}{(S-q_{C1}^{*})} x \mathbf{S}_{K}^{q_{C1^*}} \right) + \ldots\ldots$$
$$+\left(\frac{-1}{(S-q_{CN})} x \mathbf{S}_{K}^{q_{CN}} + \frac{-1}{(S-q_{CN}^{*})} x \mathbf{S}_{K}^{q_{CN^*}} \right) \qquad (17)$$

Equation (17) expresses the closed-loop transfer function as the sum of the ratio of the sensitivity of the closed-loop poles q_i over the closed-loop pole factor $(S-q_i)$, where i=1,....n, C_1,....CN, C_1*, ...CN*.

To obtain an expression for the sensitivity of a particular closed-loop pole such as q_i, multiply equation (17) by the factor $(S - q_i)$ throughout and then evaluate the result at $S = q_i$;

$$-(S - q_i)\ T_{C.L.}(S) \Big|_{S=q_i} = \mathbf{S}_{K}^{q_i} \qquad \text{All other terms cancel out,} \qquad (18)$$

$$\text{hence } \mathbf{S}_{K}^{q_i} = - r_i = - K R_i \qquad (19)$$

where r_i is the residue at the closed-loop pole q_i and R_i is the normalised residue defined as a ratio of the residue to the loop-gain K.

2.5 Sensitivity Of The Closed-Loop Pole Due To Variations Of Open-Loop Poles

The sensitivity of the ith closed-loop pole q_i due to variations of the open-loop pole P_j has been already defined in section 2.1, which is reproduced here for convenience as follows:

$$S_{P_j}^{q_i} = \frac{\delta q_i}{\delta P_j} \tag{20}$$

Partial differentiate equation (12) with respect to P_j will result into:

$$\frac{-\text{Den. open loop}}{(S-P_j)} = \frac{\delta[\text{Den. closed loop}]}{\delta P_j} \tag{21}$$

Take the natural logarithm of (Den. Closed-loop] then partial differentiate the result with respect to P_j give:

$$\frac{1}{\text{Den. closed loop}} \times \frac{\delta[\text{Den. closed loop}]}{\delta P_j} =$$

$$\frac{-1}{(S-q_1)} \times \frac{\delta q_1}{\delta P_j} + \cdots + \frac{-1}{(S-q_n)} \times \frac{\delta q_n}{\delta P_j} + \cdots + \left(\frac{-1}{(S-q_{C1})} \times \frac{\delta q_{C1}}{\delta P_j} + \frac{-1}{(S-q_{C1}^*)} \times \frac{\delta qC1^*}{\delta P_j} \right) + \cdots$$

$$+ \left(\frac{-1}{(S-q_{CN})} \times \frac{\delta q_{CN}}{\delta P_j} + \frac{-1}{(S-q_{CN}^*)} \times \frac{\delta qCN^*}{\delta P_j} \right) \tag{22}$$

Substitute equation (20) and (21) into equation (22) give:

$$\frac{1}{\text{Den. closed loop}} \times \frac{-\text{Den. open loop}}{(S-P_j)} =$$

$$\frac{-1}{(S-q_1)} \times S_{P_j}^{q_1} + \cdots + \frac{-1}{(S-q_n)} \times S_{P_j}^{q_n} + \cdots + \left(\frac{-1}{(S-q_{C1})} \times S_{P_j}^{q_{C1}} + \frac{-1}{(S-q_{C1}^*)} \times S_{P_j}^{q_{C1}^*} \right) + \cdots$$

$$\cdots + \left(\frac{-1}{(S-q_{CN})} \times S_{P_j}^{q_{CN}} + \frac{-1}{(S-q_{CN}^*)} \times S_{P_j}^{q_{CN}^*} \right) \tag{23}$$

Multiply and divide the L.H.S. of equation (23) by the factor K Num. open loop, result in the following relation:

$$\frac{1}{\text{Den. closed loop}} \times \frac{\text{-Den. open loop}}{(S-P_j)} = \frac{T_{C.L.}(S)}{(S-P_j)} \times \frac{\text{-Den. open loop}}{\text{K Num.open loop}}$$

$$= \frac{-1}{(S-P_j)} \times \frac{T_{C.L.}(S)}{T_{O.L.}(S)} \qquad (24)$$

where $T_{O.L.}(S)$ is the open-loop transfer function of the system.

To obtain an expression for the sensitivity of a particular closed-loop pole such as q_i, multiply both sides of equation (23) by the factor $(S-q_i)$ then evaluate the result at $S=q_i$, all the terms of the R.H.S. of equation (23) reduce to zero except the term containing $\mathbf{S}_{P_j}^{q_i}$, which is given as:

$$\mathbf{S}_{P_j}^{q_i} = \frac{1}{(S-P_j)} \times (S-q_i) \left. \frac{T_{C.L.}(S)}{T_{O.L.}(S)} \right|_{S=q_i} \qquad (25)$$

but $T_{O.L.}(S)\Big|_{S=q_i} = -1$ since $1 + T_{O.L.}(S) = 0$ at $S=q_i$

then

$$\mathbf{S}_{P_j}^{q_i} = \left. \frac{1}{(S-P_j)} \right|_{S=qi} \times \left[-(S-q_1) \times T_{C.L.}(S) \right] \Big|_{S=qi} \qquad (26)$$

hence $$\mathbf{S}_{P_j}^{q_i} = \left. \frac{1}{(S-P_j)} \right|_{S=qi} \mathbf{S}_{K}^{q_i}; \qquad (27)$$

with reference to equation (18). Equation (27) can, therefore, be finally reduced to:

$$\mathbf{S}_{P_j}^{q_i} = \frac{1}{(q_i-P_j)} \times \mathbf{S}_{K}^{q_i} \qquad (28)$$

2.6 Sensitivity Of The Closed-Loop Pole Due To Variations Of Open-Loop Zeros

The sensitivity of the ith closed-loop pole q_i due to variations of the open-loop zero Z_j has been already defined in section 2.1, which is reproduced here for convenience as follows:

$$S^{q_i}_{z_j} = \frac{\delta q_i}{\delta Z_j} \tag{29}$$

Partial differentiate equation (12) with respect to Z_j will result into:

$$\frac{K\delta \text{ Num. open loop}}{\delta Z_j} = \frac{\delta(\text{Den closed loop})}{\delta Z_j} \tag{30}$$

thus,

$$\frac{-K \text{ Num. open loop}}{(S-Z_j)} = \frac{\delta}{\delta Z_j}(\text{Den. closed-loop}) \tag{31}$$

But

$$\frac{\delta \text{Ln (Den closed loop)}}{\delta Z_j} = \frac{1}{(\text{Den. closed loop})} \cdot \frac{\delta(\text{Den closed loop})}{\delta Z_j} \tag{32}$$

which may be expanded as:

$$\frac{\delta \text{Ln Den. C.L.}}{\delta Z_j} = \frac{\delta}{\delta Z_j}\ [\text{Ln}(S-q_1) + \text{Ln}(S-q_2)\cdots\cdots] \tag{33}$$

This can be reduced to:

$$\begin{aligned}\frac{\delta \text{Ln Den. C.L.}}{\delta Z_j} = {} & \frac{-1}{(S-q_1)} \times \frac{\delta q_1}{\delta Z_j} + \cdots + \frac{-1}{(S-q_n)} \times \frac{\delta q_n}{\delta Z_j} + \cdots\cdots \\ & + \left(\frac{-1}{(S-q_{C1})} \times \frac{\delta q_{C1}}{\delta Z_j} + \frac{-1}{(S-q_{C1}^*)} \times \frac{\delta q_{C1}^*}{\delta Z_j}\right) + \cdots\cdots \\ & + \left(\frac{-1}{(S-q_{CN})} \times \frac{\delta q_{CN}}{\delta Z_j} + \frac{-1}{(S-q_{CN}^*)} \times \frac{\delta q_{CN}^*}{\delta Z_j}\right)\end{aligned} \tag{34}$$

Substituting equations (29), (31) and (32) into equation (34), yields:

$$\frac{1}{\text{Den. closed loop}} \times \frac{\text{-K Num. open loop}}{(S-Z_j)} =$$

$$\frac{-1}{(S-q_1)} x \mathbf{S}^{q_i}_{z_j} + \cdots + \frac{-1}{(S-q_n)} x \mathbf{S}^{q_n}_{z_j} + \cdots + \left(\frac{-1}{(S-q_{C1})} x \mathbf{S}^{q_{C1}}_{z_j} + \frac{-1}{(S-q_{C1}^{*})} x \mathbf{S}^{q_{C1*}}_{z_j} \right)$$

$$+ \cdots + \left(\frac{-1}{(S-q_{CN})} x \mathbf{S}^{q_{CN}}_{z_j} + \frac{-1}{(S-q_{CN}^{*})} x \mathbf{S}^{q_{CN*}}_{z_j} \right) \qquad (35)$$

To obtain an expression for the sensitivity of a particular closed-loop pole such as q_i, multiply both sides of equation (35) by the factor $(S-q_i)$ then evaluate the result at $S=q_i$, thus:

$$\mathbf{S}^{q_i}_{z_j} = \frac{-(S-q_i)}{\text{Den. C.L.}} x \frac{\text{-K Num. open loop}}{(S-Z_j)} \Big|_{S=q_i} \qquad (36)$$

which can be re-written as:

$$\mathbf{S}^{q_i}_{z_j} = \frac{(S-q_i)}{(S-Z_j)} \cdot T_{C.L.}(S)\Big|_{S=q_i} \qquad (37)$$

Using equation (18); equation (37) becomes

$$\mathbf{S}^{q_i}_{z_j} = \frac{-1}{S-Z_j} \Big|_{S=q_i} x \mathbf{S}^{q_i}_{K} \qquad (38)$$

or

$$\mathbf{S}^{q_i}_{z_j} = \frac{-1}{q_i-Z_j} x \mathbf{S}^{q_i}_{K} \qquad (39)$$

2.7 Foundation Of A Novel Sensitivity-Based Algorithm

For unity feedback SISO systems, three inter-linked equations have been derived in the previous sections which relate the sensitivity of any closed-loop pole q_i due to small changes either in the loop gain K (equation 19), in an open-loop pole Pj (equation 28), or in open-loop zero Z_j(equation 39). The interlinking function has been shown to be the normalised residue of the closed-loop pole R_i.

The effect of the change in any single or in any combination of these parameters is reflected in migrations of the closed-loop poles away from their original positions.

For instance, the migration vector Δq_i of the ith closed-loop pole q_i due to ΔK change in the loop gain K can be derived from equation(11) and (19) to give approximately:

$$\Delta q_i \cong \frac{\Delta K}{K} \times [-KR_i] = -R_i \times \Delta K \tag{40}$$

$$\text{Where } \mathbf{S}_{\mathbf{K}}^{\mathbf{q_i}} = \frac{\delta q_i}{\delta K/K} = \text{Lim} \frac{\Delta q_i}{\Delta K/K} = -KR_i \tag{41}$$

Equation (40) is only accurate when ΔK is infinitesimally small, otherwise R_i cannot be assumed to remain constant during that change.

Similarly, the migration vector Δq_i of the ith closed-loop pole q_i due to ΔP_j change in the jth open-loop pole P_j can be derived from equations (20) and (28) to give approximately:

$$\Delta q_i \cong \Delta P_j \times \frac{[-KR_i]}{(q_i - P_j)} = [-KR_i] \times \frac{\Delta P_j}{q_i - P_j} \tag{42}$$

$$\text{where } \mathbf{S}_{\mathbf{P_j}}^{\mathbf{q_i}} = \frac{1}{q_i - P_j} \cdot \mathbf{S}_{\mathbf{K}}^{\mathbf{q_i}} = \frac{1}{q_i - P_j} \times [-KR_i] \tag{43}$$

Again, the accuracy of equation (42) can only be guaranteed if ΔP_j is infinitesimally small so that R_i can be assumed to remain constant during the change.

In the same way, the migration vector Δq_j of the ith closed-loop pole q_i due to ΔZ_j change in the jth open-loop zero Z_j can be deduced from equations (29) and (39) to give approximately:

$$\Delta q_i = -\Delta Z_j \times \frac{[-KR_i]}{(q_i - Z_j)} = [-KR_i] \times \frac{-\Delta Z_j}{q_i - Z_j} \tag{44}$$

$$\text{where } \mathbf{S}_{Z_j}^{q_i} = \frac{-1}{(q_i - Z_j)} \cdot \mathbf{S}_{K}^{q_i} = \frac{-1}{(q_i - Z_j)} \times [-KR_i] \tag{45}$$

A similar argument regarding the accuracy of equation (44) still applies as that advanced for equations (40) and (42).

It follows , therefore, from the above derived equations, that the total migration vector Δq_i of the ith closed-loop pole q_i due to concurrent changes of ΔK, ΔPj and ΔZ_j in the open-loop gain and in the locations of the open-loop poles and open-loop zeros, respectively, can be obtained approximately as:

$$\Delta q_{i\ \text{total}} \cong -R_i \left(\Delta K + \sum_{j=1}^{j=n_1} \frac{K\,\Delta P_j}{(q_i - P_j)} + \sum_{j=1}^{j=n_2} \frac{-K\,\Delta Z_j}{(q_i - Z_j)} \right) \tag{46}$$

where n_1 represents all the varying open-loop poles, and

n_2 represents all the varying open-loop poles.

And the total sensitivity of q_i as:

$$\mathbf{S}_{(K,\,P_j,\,Z_j)}^{(q_i\quad)} = \mathbf{S}_{K}^{q_i} + \sum_{j=1}^{j=n_1} \mathbf{S}_{P_j}^{q_i} + \sum_{j=1}^{j=n_2} \mathbf{S}_{Z_j}^{q_i}$$

$$\mathbf{S}_{(K,\,P_j,\,Z_j)}^{(q_i\quad)} = [-KR_i] \left(1 + \sum_{j=1}^{j=n_1} \frac{1}{(q_i - P_j)} + \sum_{j=1}^{j=n_2} \frac{-1}{(q_i - Z_j)} \right) \tag{47}$$

The set of six equations (40) to (45) represent the primary functional relationships on which the new algorithm is based. The set comprises three subsets of twin equations (40 & 41), (42 & 43) and (44 & 45) which mirror the three types of plant parameter variations, namely: loop gain, open-loop pole and open-loop zero variations, respectively. Each subset relates explicitly each of the systems roots as equal to the negative value of its residue, on one hand, and predicts the incremental victor displacement of the root as proportional to both the residue and the step size of the parameter variations, on the other. These twin relationships effectively indicate, therefore, that the knowledge of the root-residues would allow concurrent; hence efficient estimation of the system's sensitivity (which, in turn, is a measure of the system robustness), and the prediction of the new positions of the closed-loop poles (which reveal the system's performance and its time-domain response).

It follows, therefore, that by considering a series of successive (and relatively small) steps of a parameter variation, it is possible to build and display graphically (in 3-d) a composite performance-sensitivity profile of the system in the S- domain (in the form of combined root- loci and root-sensitivity loci), which highlights, continuously and effortlessly, the system behaviour, with the choice of viewing it over a wide range and at a convenient scale.

It can also be seen from the duality of these subsets of twin relations, that the composite profile of the system behaviour can be easily constructed for combinations of any two types of parameter variations having independent rates. Extending this further, the set of twin equations (46) & (47), for instance, represents the integration of the above three subsets with which the composite profile of the system behaviour can be generated, when the three types of parameter variations take place with any relative rates, as manifested indirectly by the relative step sizes of the ΔK, ΔP_j, and ΔZ_j, respectively.

2.8 Adaptation Of The Algorithm For Computer Estimation Of Composite Root-Sensitivity And Root-Migration Contours

2.8.1 Computation Of The Initial Values Of The Root-Sensitivity

The previously derived equations (41), (43) and (45), or indeed any of their combinations, determine the exact values of the root-sensitivity for a particular loop gain K provided that the normalised residue R_i is known as a priori. The residue of any closed-loop pole, such as q_i, can however be calculated provided that the locations of all the closed-loop poles and zeros, i.e., the closed-loop transfer function $T_{C.L.}(S)$ are known for the loop gain K. Initially when K=0, this information is known if the open-loop transfer function were known; since

$$\lim_{K\to 0} T_{C.L(S)} = T_{O.L(S)} \tag{48}$$

as can be deduced from equation (12). Equation (48) implies that while the closed-loop zeros are identical to their corresponding open-loop (valid for all values of K for unity-feedback systems), initially the closed-loop poles are also identical to their corresponding open-loop, i.e.,

$$q_i\big|_{K=0} = P_i \qquad \text{for all values of i} \tag{49}$$

It follows, therefore, that $R_i\big|_{K=0}$ can be calculated as follows:

$$R_i\big|_{K=0} = \frac{1}{K}(S-P_i)T_{O.L.}(S)\big|_{S=P_i \,\&\, K=0} \tag{50}$$

where K cancels out with that existing in the numerator of $T_{O.L.}(S)$ [see equation (2)], therefore,

$$R_i\big|_{K=0} = \frac{\text{Product of vectors from open-loop zeros Zj to } P_i}{\text{Product of vectors from the rest of the open-loop poles Pj to } P_i} \tag{51}$$

For the majority of cases, $R_i|_{K=0}$ is finite, therefore, the initial value of the sensitivity of the closed-loop pole q_i is generally:

$$\mathbf{S}_{\mathbf{K}}^{q_i}\Big|_{K=0} = 0 \qquad (52)$$

Computation of equation (51) is straight forward provided P_i is a distinct pole. For the case of a multiple pole of order, say m, this should first be replaced approximately by an equivalent cluster of m distinct poles (P_{i1},.....P_{im}), uniformly distributed on the circumference of a circle centred at the position of P_i (and symmetrically about its horizontal diameter to ensure physical realisability), with an infinitesimal radius ε_i, as explained in section 2.3. In this way, equation (51) can be used as a standard routine throughout, provided that the computation procedure incorporates the following:

1. an automatic method of detecting the presence of one or more high-order poles, their locations and their orders from the initial data; namely $T_{O.L.}(S)$ that is fed in,
2. an initial default value for ε_i which may be reduced automatically when the resulting errors exceed some predetermined value according to an appropriate error criterion, and
3. a method of estimation of the positions of the distinct poles (P_{i1},.....P_{im}) to replace the mth order pole at P_i, and repeating the same for other high-order poles.

While procedure 1 is best left to the software implementation stage, as will be discussed in the next chapter, procedures 2 & 3 are more appropriately discussed here:

In the vicinity of an mth order pole at P_i, it is possible to prove that the m branches of the root-locus emanating from the position of P_i are approximately straight lines, uniformly inclined to the horizontal by the angle θ_{i1}, ... θ_{im}, where the inter-angle

between any two successive branches is equal to $2\pi/m$, and θ_{i1} depends on the pattern of the rest of the poles and zeros of $T_{O.L.}(S)$. Focusing on the local circular region centred on P_i as shown in figure 2.3; the smaller the radius ε_i, the closer the multiple branches of the root-locus to straight lines which intersect the circumference at $P_{i1},.....P_{im}$, therefore, the value of ε_i should be much smaller relative to **the smallest inter-distance between any pairs of pole/zero of $T_{O.L.}(S)$, "min-interdistance"** which can be readily calculated from the fed-in data. Therefore, it was reasonable to select the initial default value of ε_i such that:

$\varepsilon_{i\ default}$ =0.01 x min-interdistance

whereupon, the locations of $P_{i1},.....P_{im}$ will be in general:

$$P_{im} = P_i + \varepsilon_{i\ default}\,[\cos\theta_{mi} + j\sin\theta_{mi}]\ ;\ m_i = 1, 2. .... m \qquad (53)$$

which represent the points of intersection between the m root-locus branches and the circumference of the ε_i-circle. Since $P_{i1},.....P_{im}$ lie on the root-locus, they must satisfy the **angle criterion;** that the sum of the angles of the vectors from all the zeros of $T_{O.L.}(S)$ to P_{i1} minus the sum of the angles of the vectors from all the poles of $T_{O.L.}(S)$ to P_{i1} is equal to $\pm\pi$. Since the sum of the angles of the vectors from the m poles at P_i to P_{i1}, is equal to $-m\theta_{i1}$, therefore

$$\theta_{i1} = \frac{\left(\begin{array}{c}\pm\pi * (2m_i-1) + \Sigma \text{ angles of vectors from all the zeros of} \\ T_{O.L.}(S) \text{ to } P_{i1} - \Sigma \text{ angles of vectors from all the poles} \\ \text{of } T_{O.L.}(S) \text{ less the mth order pole at } P_i \text{ to } P_{i1}\end{array}\right)}{m};\ m_i = 1 \qquad (54)$$

For the special case of $\theta_{i1}=0, \pm 2\pi, ...$, however, the root-locus branches; hence the replacement distinct poles $P_{i1},.....P_{im}$ lie on the real axis of the S-plane. When this case is detected, equation (53) will no longer be valid, and the positions of the replacement distinct poles $P_{i1},.....P_{im}$ should be simply selected as:

$$P_{i1},.....P_{im} = P_i \pm \frac{2\varepsilon_{i\ default}}{m_i},\ \text{where } m_i=2, 4, ... m;\ m \text{ is even.} \qquad (55)$$

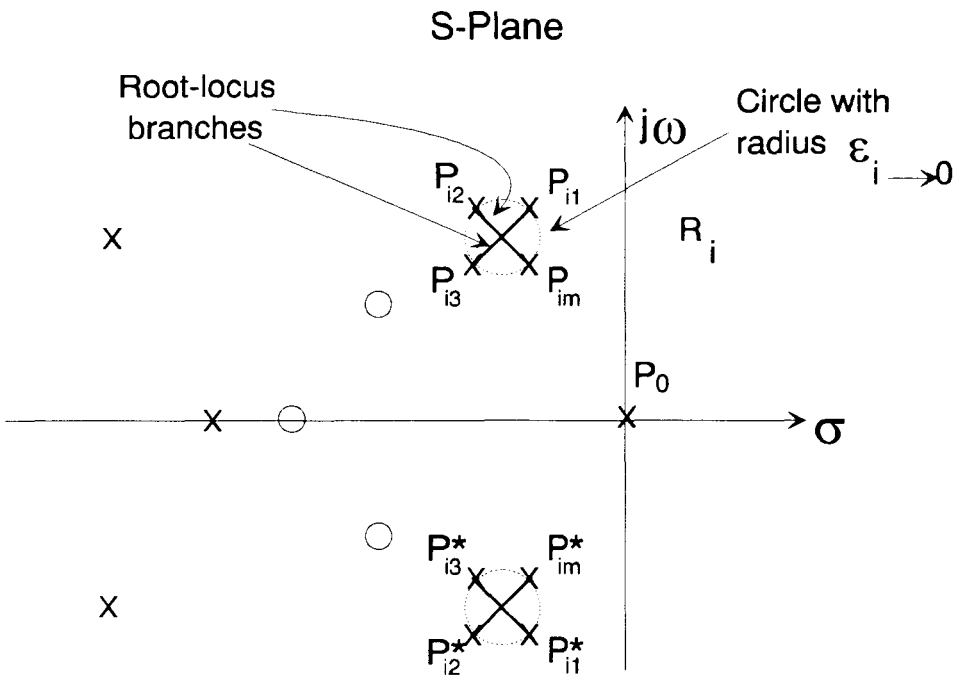


Figure 2.3 Illustration Of Method Of Replacing An mth Order Open-Loop Pole P_i By An Equivalent Cluster Of m Distinct Poles On The Root-Locus Branches, At A Distance Of $\varepsilon_i \longrightarrow o$ From P_i, Prior To Systematic Computation Of The Residues.

2.8.2 Prediction Of The Closed-Loop Poles Migration Contour

The second member of each of the three twin subset equations, namely (40), (42) and (44), estimates the anticipated incremental migrations of the closed-loop poles in response to a small variation in the loop gain, in the open-loop pole (or poles), or in the open-loop zero (or zeros), respectively. During an incremental step variation of size ΔK, ΔP_j, and ΔZ_j, the values of the corresponding normalised residues R_i and the loop gain K ($K= \Sigma \Delta K$ over the number of steps) are expected to vary slightly. In addition, both q_i & P_j or q_i & Z_j are also expected to vary slightly in equations (42) and (44), respectively. Therefore, it is **vital** that a very small step size ΔK, ΔP_j, and ΔZ_j should be selected in order to minimise the accumulative errors.

One way of achieving this is first to select initial default values for each of ΔK, ΔP_j, and ΔZ_j at the beginning of each computational cycle based on some reasonable assumptions. Secondly, at the end of each computational cycle, the deviation of each closed-loop pole from the locus along which it should migrate, will be computed and compared with a predetermined tolerable value according to an error criterion. In the case of violation of the criterion, the same computational cycle is automatically re-activated after reducing the initial default value of the step size by half. In principle, this process may repeat as many time, each time the step size is reduced again by half of its previous value, until the error criterion is satisfied.

Adapting the above procedure, the following parameters will have been updated at the end of the Nth step:

$$R_{i(N)} = \frac{1}{K_{(N-1)}} \left(S - q_{i(N-1)} \right) T_{C.L.(N-1)}(S) \Big|_{S=q_{i(N-1)}} \qquad (56)$$

$$K_{(N)} = K_{(N-1)} + \Delta K_{(N)} \quad ; \; K_{(0)} = 0 , \qquad (57)$$

$$P_{j(N)} = P_{j(N-1)} + \Delta P_{j(N)} \quad ; \; P_{j(0)} = P_j , \qquad (58)$$

$$Z_{j(N)} = Z_{j(N-1)} + \Delta Z_{j(N)} \quad ; \; Z_{j(0)} = Z_j , \qquad (59)$$

and

$$q_{i(N)} = q_{i(N-1)} + \Delta q_{i(N)} \quad ; \; q_{i(0)} = P_{i(0)} , \qquad (60)$$

after satisfying the angle criterion that:

Sum of the angles of vectors from all the updated open-loop zeros $Z_{j(N)}$ to the updated closed-loop pole $q_{i(N)}$ - sum of the angles of vectors from the updated open-loop poles $P_{j(N)}$ to the updated closed-loop pole $q_{i(N)} = \pm \pi \pm epsi_{(N)}$ (61)

where $epsi_{(N)}$ is the updated tolerable error in the angle, which is dependant on the resolution of the current section of the root-locus being computed as well as on the

application under consideration. i.e., $epsi_{(N)}$ is dependant on $\Delta K_{(N)}$, $\Delta P_{j(N)}$ and $\Delta Z_{j(N)}$. Therefore, $epsi_{(N)}$ was appropriately selected as:

$$epsi_{(N)} \leq 0.001 \frac{\left(\Delta K_{(N)} + \Delta P_{j(N)min} + \Delta Z_{j(N)min} \right)}{1 + \text{No. of varying O.L. poles } n_1 + \text{No. of varying O.L. zeros } n_2} \tag{62}$$

wherein

$$\Delta K_{(N)} = \frac{0.01 \text{ min-inter-distance between any pair of pole/zero of } T_{C.L.(N-1)}(S)}{\left| R_{i(N-1)max} \right|} \tag{63}$$

$$\Delta P_{j(N)min.} = \frac{\text{minimum total anticipated change in all } P_j}{\text{Total number of steps}} \tag{64}$$

provided $\Delta P_{i(N)} \leq \Delta K_{(N)}$, otherwise double the total number of steps and repeat equation (64) until $\Delta P_{j(N)min.}$ is satisfied.

$$\Delta Z_{j(N)min.} = \frac{\text{minimum total anticipated change in all } Z_j}{\text{Total number of steps}} \tag{65}$$

provided $\Delta Z_{i(N)} \leq \Delta K_{(N)}$, otherwise double the total number of steps and repeat equation (65) until $\Delta Z_{j(N)min.}$ is satisfied.

The fractions 0.001 and 0.01 appearing in equations (62) and (63), respectively, represent initial default figures.

Therefore, the estimated total incremental migration value of the ith closed-loop pole at the end of the step (N) can be calculated through application of equation (46) as follows:

$$\Delta q_{i\ total(N)} \cong -R_{i(N)} \left(\Delta K_{(N)} + \sum_{j=1}^{j=n_1} \frac{K_{(N)} \Delta P_{j(N)}}{(q_{i(N)} - P_{j(N)})} + \sum_{j=1}^{j=n_2} \frac{-K_{(N)} \Delta Z_{j(N)}}{(q_{i(N)} - Z_{j(N)})} \right) \tag{66}$$

and the corresponding total sensitivity (at the beginning of step (N+1), through application of equation (47) as follows:

$$\mathbf{S}^{(q_i)}_{(K,\, P_j,\, Z_j)_{(N+1)}} \cong - K_{(N)}\, R_{i(N)} \left(1 + \sum_{j=1}^{j=n_1} \frac{1}{(q_{i(N)} - P_{j(N)})} + \sum_{j=1}^{j=n_2} \frac{-1}{(q_{i(N)} - Z_{j(N)})} \right) \tag{67}$$

For the case of no variations in the open-loop poles, the second term (between the large brackets) of equations (66) and (67) will reduce to zero.

Likewise, for no variations in the open-loop zeros, the third term (between the large brackets) of equations (66) and (67) will reduce to zero.

The justification of the arbitrary division by the modulus of the largest normalised residue $\left| R_{i(N-1)max.} \right|$ in equation (63) for the determination of $\Delta K_{(N)}$ is that it would result in constant incremental migration steps of $\Delta q_{i\ total(N)}$, along any branch of the root-contour, as can be deduced from equation (40). As a consequence, viewing and analysis of the resulting composite graphics are enhanced.

2.8.3 Detection Of Saddle Points Along The Closed-Loop Poles Migration Contour And Computation Strategy In Their Vicinity Areas

As pointed out earlier, the values of the residues; hence the sensitivities would rise very sharply as two or more of the closed-loop poles cross each other path at saddle points. Theoretically, these values reach infinity. Therefore, it is expected that computation problems such as overflow errors will arise in the vicinity of the saddle points, regardless of the direction of the closed-loop poles migration whether towards or away

from them. It is essential, therefore, that the regions close to the saddle points should be avoided without the loss of the continuity of the sequential computation arrangements of the pervious section nor their accuracy outside these regions.

To achieve these goals, the following measures are required:

1. Introduction of an automatic means of predicting the saddle points along the closed-loop poles migration contour, and a means of locating their positions accurately.

2. Incorporation of an automatically activated decision-making procedure for the most appropriate size of circular zones surrounding the saddle points (referred to subsequently as the forbidden zones), inside which no computation is permitted, without compromising the accuracy nor the continuation of computation outside them.

3. Incorporation of a means of finding the points of their departure from the circumference of the forbidden zones at which a boundary computation procedure will have to be invoked before reverting back to the previous sequential procedure.

For saddle point detection, the inter-distances between every two pairs of updated closed loop poles' positions at the beginning of each computation cycle are measured. For instance, in the beginning of (N+1)th cycle, $q_{i(N+1)}$ [see equation (60)] are available which allow the computation of these inter-distances. These are then compared sequentially with the modulus of the predicted values of the incremental migration distance $\left| \Delta q_{i\ total(N+1)} \right|$. If two or more of these inter-distances-are less than or equal to $\left| \Delta q_{i\ total(N+1)} \right|$, then a saddle point is predicted and the relevant closed-loop poles are grouped together and assigned a code for further recall and processing. The process of these comparisons then proceeds for the rest of the closed-loop poles to find whether other saddle points can also be detected, whereupon the relevant closed-loop poles are similarly grouped together and each assigned a different code to allow discrimination between the groups. The position of the saddle point of

each group is then determined as the centre of gravity of the relevant group of the closed-loop poles, which are now assumed to lie on the circumference of the forbidden zone at the points of entry which theoretically should be uniformly distributed as shown in the illustration of figure 2.4. The segments of the migration-locus pointing towards the saddle point within the forbidden circle are expected to be nearly straight lines, so also are the segments of the emerging migration locus, bisecting the angles between every two successive entry points.

Therefore, the points of departure on the circumference of the forbidden circle can be determined via the angles of entry, angles of departure, the radius of the circle and the position of the saddle point. These points of departure represent the updated positions of the closed-loop poles of this particular group at the end of the computation cycle (N+1). Similarly, the updated positions of the closed-loop poles belonging to other groups can also be obtained. To calculate the updated positions of the rest of the closed-loop poles which do not cross at a saddle point, they can be assumed to incrementally migrate along their respective loci during (N+1)th step by an incremental vector having:

> a length approximately equal to twice the mean radius of all the forbidden zones, and approximately in the same direction as that of the previous incremental migration at step (N).

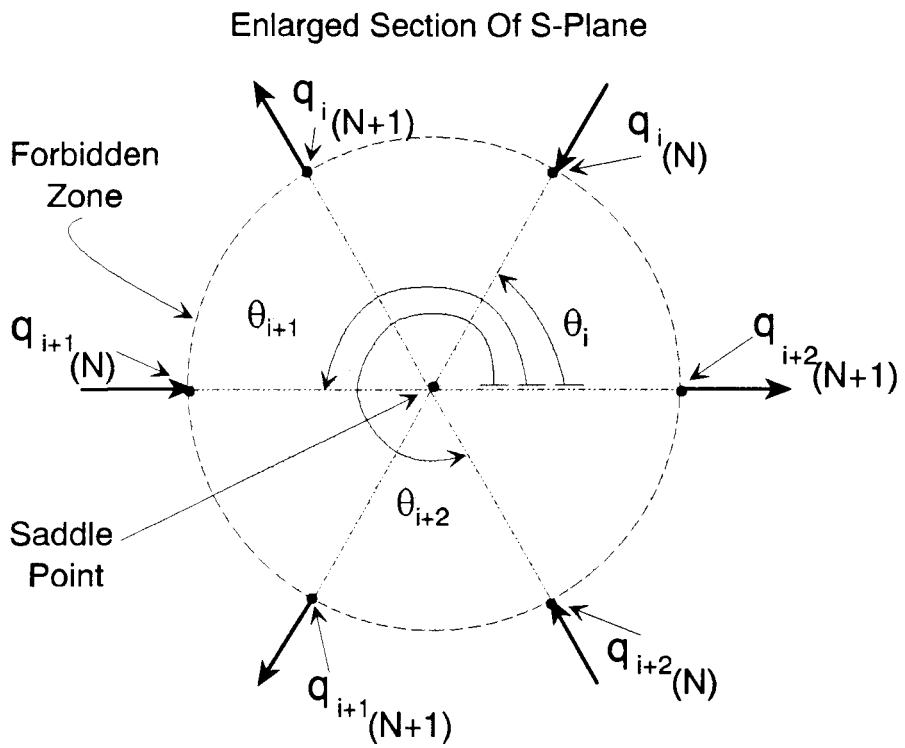


Figure 2.4 Illustration Of The Details For Detection Of Saddle Point And For Prediction Of The Closed-Loop Poles' Positions On The Emerging Migration Contour Loci

Having determined $q_{i(N+1)}$, the updated normalised residues $R_{i(N+1)}$ can be calculated as follows:

$$R_{i(N+1)} = \frac{1}{K_{(N+1)}} (S-q_{i(N+1)}) \; T_{C.L.(N+1)}(S) \Big|_{S=q_{i(N+1)}} \tag{68}$$

Although the value of $K_{(N+1)}$ is not required in the evaluation of equation (68) since it cancels out due to its appearance in the numerator of $T_{C.L.(N+1)}(S)$, however, in order to re-initiate a sequential set of recurrent equations similar to equation (56) to (67), $K_{(N+1)}$, $P_{j(N+1)}$ and $Z_{j(N+1)}$ are required to predict the migrations of the closed-loop poles after emerging from the forbidden zones. These parameters can be obtained approximately as follows:

$$P_{j(N+1)} \cong P_{j(N)} + \Delta P_{j(N)} \quad ; \; \Delta P_{j(N)} \cong \Delta P_{j(N-1)}, \tag{69}$$

$$Z_{j(N+1)} \cong Z_{j(N)} + \Delta Z_{j(N)} \quad ; \; \Delta Z_{j(N)} \cong \Delta Z_{j(N-1)}, \tag{70}$$

and $K_{(N+1)}$ can be obtained by invoking the magnitude criterion of the root-locus, thus:

$$K_{(N+1)} \cong \left| \frac{\begin{pmatrix}\text{Product of the vectors from the updated open-loop poles} \\ P_{j(N+1)} \text{ to one of the updated closed-loop pole } q_{i(N+1)}\end{pmatrix}}{\begin{pmatrix}\text{Product of the vectors from the updated open-loop zeros} \\ Z_{j(N+1)} \text{ to one of the same updated closed-loop pole } q_{i(N+1)}\end{pmatrix}} \right|$$

For future computation steps beyond (N+1), the algorithm reverts back to the sequential set of recurrent equations (56) to (67) provided no more saddle points were predicted, otherwise, the above sequence of procedures including equations (68) to (71) are invoked.

2.8.4 Accuracy-Enhancement Scheme For The Sensitivity-Based Algorithm

The previous section has mapped out the mechanisation processes necessary for converting the algorithm into a format suitable for automated numerical computations. Concurrently, it has highlighted the procedure for evaluating the accuracy and the error-criterion adopted to force the algorithm automatically to perform within a predetermined resolution. This was, however, achieved at the expense of speed of computation since some of the computation cycles may have to be repeated several times with successively reduced step sizes of $\Delta K_{(N)}$, $\Delta P_{j(N)}$ and/or $\Delta Z_{j(N)}$.

The objective of this section is, therefore , to introduce a scheme for reducing the errors of estimation within each step so that the accumulative errors over a number of steps would still be at a sufficiently low level to trigger the action of the error-criterion except for certain rare occasions.

The scheme divides each step cycle (excluding the cycles involving the saddle points) into three substeps as described below for the Nth cycle:

Substep 1

Invoking the sequence of computations by following the recurrent equations (56) to (67), using the values of the parameters:

$q_{i(N-1)}$, $T_{C.L.(N-1)}(S)$, $K_{(N-1)}$, $P_{j(N-1)}$ and $Z_{j(N-1)}$, obtained at the end of the (N-1)th step cycle, will ultimately culminate in the estimation of:

$$q_{i(N)1} \text{ and } \mathbf{S}^{q_i}_{(K,\, P_j,\, Z_j)_{(N)1}}$$

Since the estimated incremental migration of the ith closed-loop pole $\Delta q_{i\ total(N)1}$ (which represents approximately a tangent to the root-contour at $q_{i(N-1)}$) inherently contains small errors, as explained earlier in this chapter; the updated position of that pole is expected to deviate from the actual root-contour as shown in figure 2.5.

Substep 2

Repeating substep 1, using the updated relevant parameters obtained at the end of the substep1 will give the estimation of $q_{i(N)2}$ and $\mathbf{S}^{q_i}_{(K,\, P_j,\, Z_j)_{(N)2}}$

Now the cumulative errors contained in the estimated incremental migration of the ith closed-loop pole $\Delta q_{i\ total(N)2}$ will cause a relatively larger deviation of the updated position of that pole from the actual root-contour.

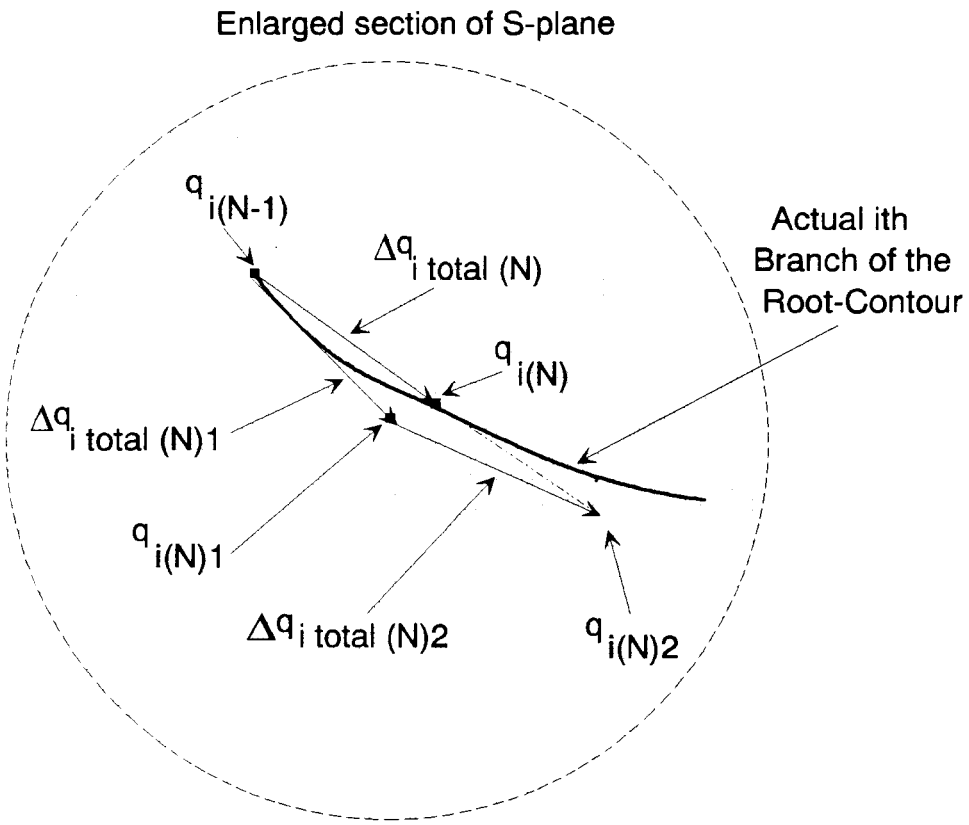


Figure 2.5 Illustration Of The Details Of The (N)th Three Sub-Step Cycle Of The Enhanced Algorithm For Predicting The New Position Of The ith Closed-Loop Pole $q_{i(N)}$.

Substep 3

This is designed primarily to compensate partially for the cumulative errors of the previous two substeps by deducing a better estimate of the incremental migration of the closed-loop pole. This is obtained by taking the mean value of this parameter, obtained in substep 1 and substep 2, as follows:

$$\Delta q_{i(N)3} = 0.5[\Delta q_{i(N)1} + \Delta q_{i(N)2}]$$
$$= \Delta q_{iN} \qquad (72)$$

The effect of the above process is best explained by the diagram of figure 2.5, where the vectorial summation $[\Delta q_{i(N)1} + \Delta q_{i(N)2}]$ gives a resultant vector $[q_{i(N)1} - q_{i(N)2}]$ whose mid point, which represents the adopted updated position of the ith closed-loop

pole at the end of the (N)th step cycle, lies closer to the actual root-contour compared to either $q_{i(N)1}$ or $q_{i(N)2}$.

The above method is well established technique employed widely for tracking state-space trajectories with greater accuracy.

2.9 Conclusions

The work in the first part of this chapter has been targeted to the derivation of the fundamental formulations for a new versatile CAD algorithm.

It has been shown that the residues played the key role in interfacing the root-sensitivity with the root-migration contour. This theoretical framework establishes a highly efficient environment for automated concurrent prediction of feedback systems behaviour subject to all types of uncertainty, noise and/or experiencing a diverse range of faults, as will be discussed in the later part of this thesis. The most significant features of the algorithm development, however, lie in its ability to measure simultaneously performance and quality assurance (e.g., robustness and reliability) as projected in different domains (S-domain, Z-domain, frequency-domain and time-domain) with great speed and accuracy; which represent the primary ingredients for a versatile tool for both analysis and design.

The work in the second part of the chapter has focused on the adaptation of the algorithm formulations for numerical computations prior to software implementation. Here, details of automated procedures, solutions to computational problems such as near the saddle points and higher-order poles, insertion of appropriate error-criteria, default values and guide to user's over-ride, and a scheme for accuracy enhancement have been outlined.

Although the work relates to single-input single-output feedback systems, the algorithm has been designed with an in-built flexible structure from the outset so as to allow its extension to handle multi-input multi-output systems and the integration of future computer advances such as interactive graphics.

Primarily, the work has highlighted an universal adaptive CAD approach to the analysis and the design problems associated with widely diverse applications operating in a real environment.

Chapter 3

Software Implementation Of The S-B Algorithm As A Universal CAD System

3.1 Introduction

The sensitivity-driven formulations derived in Chapter 2 together with their detailed adaptations are now ready for translation into a software implementation.

It has been shown that the algorithm basically predicts the composite root-migration contour root-sensitivity profile for SISO feedback systems of any order and of any pole/zero configuration in general. Different configurations, however, may require different approach to the algorithm implementation under certain circumstances, e.g., presence of higher-order open-loop poles and/or saddle points on the root-migration contour. Furthermore, the feedback system may contain a plant subject to parameter perturbations, noise, or indeed fault conditions. The implementation should, therefore, reflect all these aspects through the incorporation of automatic identification procedures, criteria for self evaluation together with appropriate user-machine interactions. The work in chapter 2 has largely dealt with these issues or at least paved the way towards feasible solutions. More and above, two important issues have to be resolved namely: accuracy and ambiguities.

At the end of chapter 2, a scheme has been designed to address the first issue. The second issue. however, by their very nature will only arise during the implementation stage, whereupon automatic diagnostic procedures and provision of appropriate routines have yet to be devised.

The aim of this chapter is, therefore, to present a coherent approach to the software implementation of the S-B algorithm as a fully fledged universal CAD system. The implementation environment has been chosen:

* targeted to a personal computer (IBM PC or its compatible) operating under MS-DOS, as it is widely available,

* portability, easy to modify and update by using the high-level language Turbo-C (Borland international product [58]).

Apart from the incorporation of all the above aspects, the implementation structure integrates 2-d and 3-d graphic facilities with provision of: defaults, zoom, rotational perspectives option, fast and slow displays of continuous parameter modifications as well as user over-ride options, screen windows and command menus.

The main modules of the CAD implementation are: inputs, outputs, data handling and processing, decision making, parameter selections, angle calculations and error checking, root-sensitivity and closed-loop pole migration calculations, high-order pole identification, saddle point detection and processing, test for physical realisability, modelling of parameter uncertainty, and fault simulation.

The complete listing of the S-B CAD system program is given in appendix A.

3.2 The Structure Of The S-B CAD System

Figure 3.1 shows the block diagram of the structure of the S-B CAD system implementation showing the high-level interactions between the CAD system and the outside environment as well as between the various processing blocks. Figure 3.2, on the other hand, focuses on the functions of the blocks and their interactions as outlined below:

3.3 Initialisations and Declarations

The various subroutines of the S-B program implementation are initialised by the general Turbo-C programming data and functional declarations, and through graphic devices set-up, assignment of memory allocations to arrays and resetting their working areas to zero (to prevent data corruption, e.g., random numbers may already be present in the arrays, on powering up).

3.4 Inputs To The S-B CAD System

The system requires a number of inputs that can either be entered via the keyboard, file transfer, or through an interface device from another machine. The types of the inputs are defined as follows:

(i) The window size in the S-plane in terms of the number of steps of the loop cycles to be executed.

(ii) The number of the open-loop poles and zeros in complex format.

(iii) Either the default value or the user option of ΔK in terms of the selection of the magnification/attenuation factor $epsi_i$.

(iv) The plant delay.

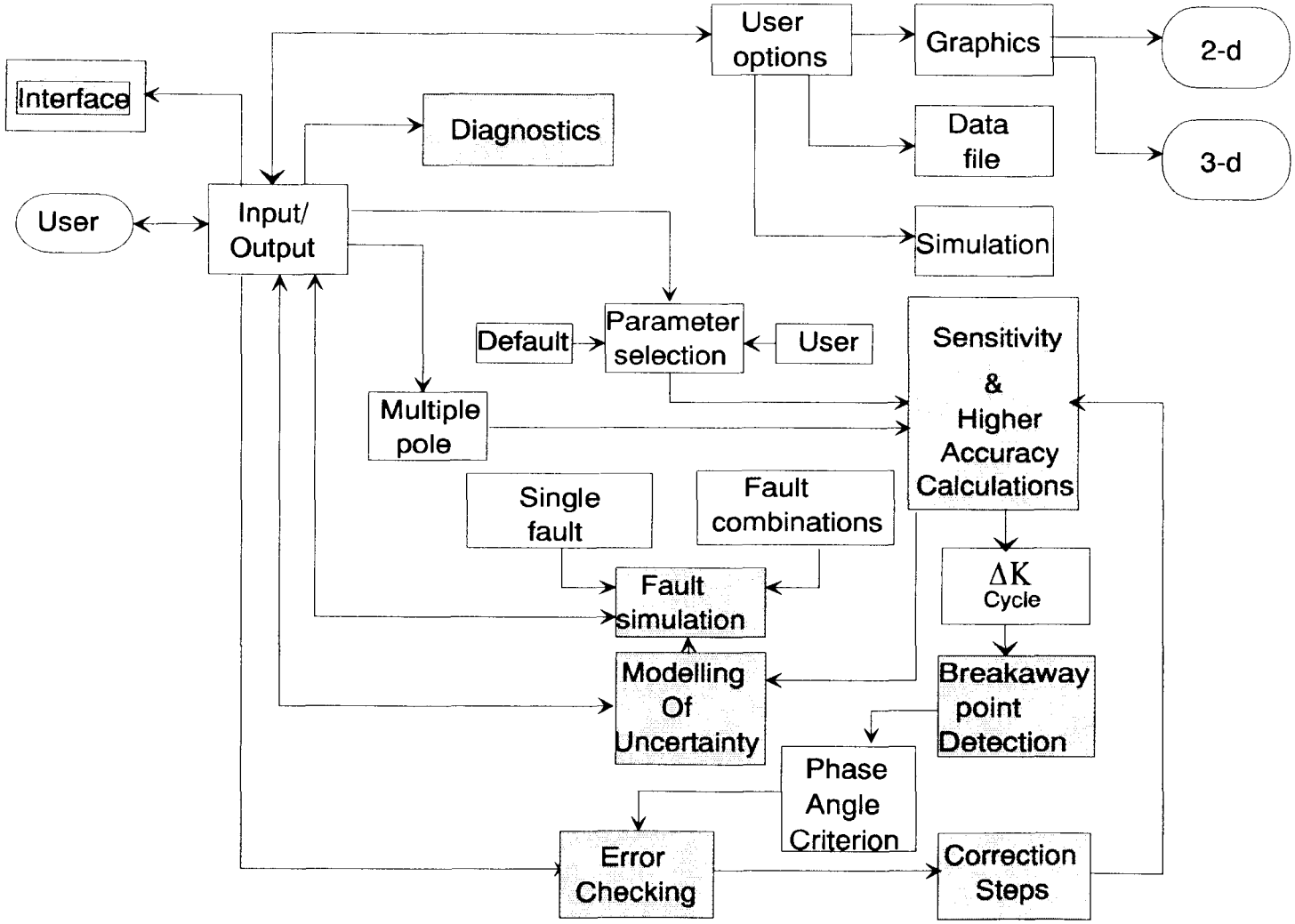


Figure 3.1 High-level Block Diagram Of The S-B CAD System Implementation

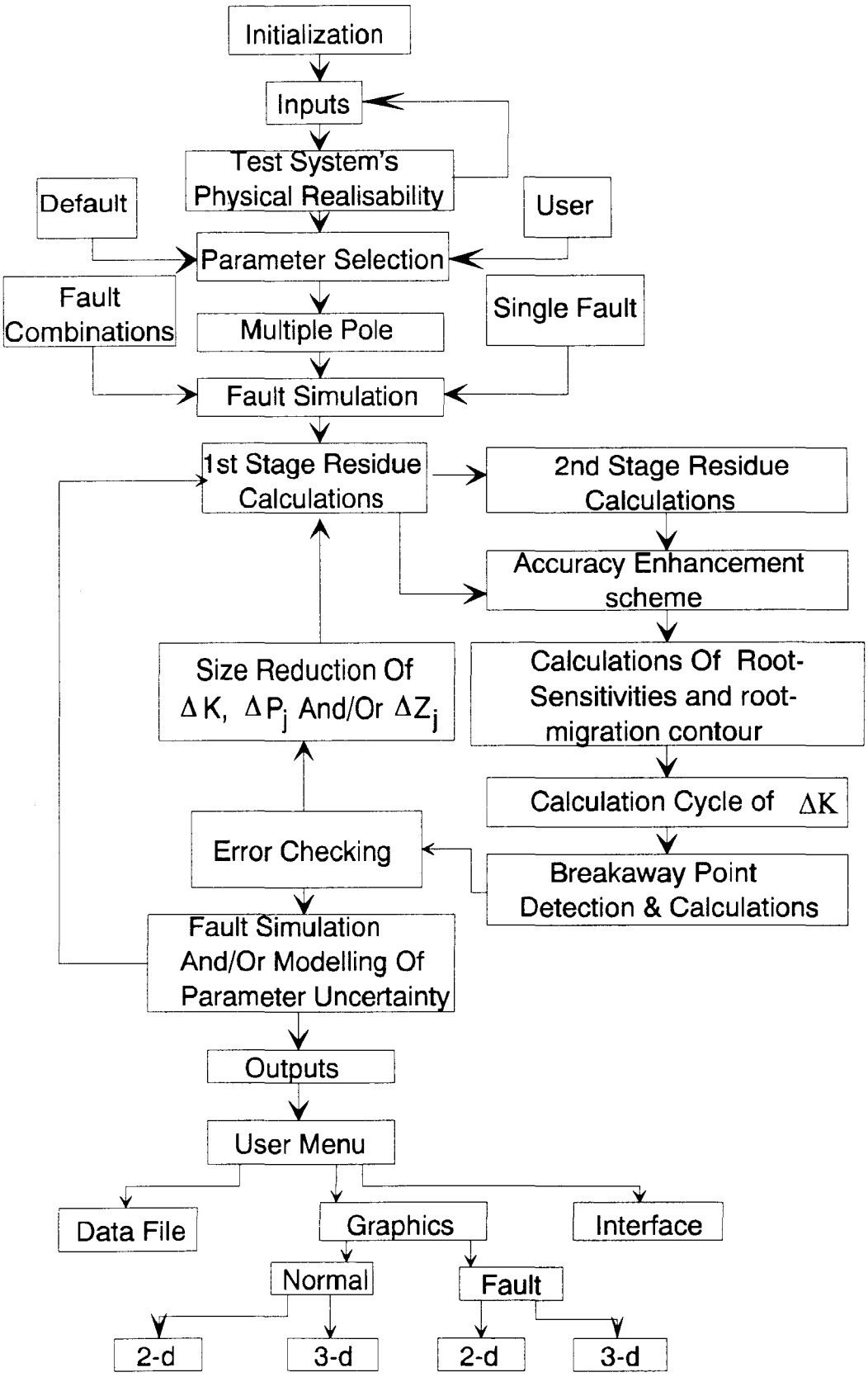


Fig. 3.2 Functional Representation Of The S-B CAD System Structure

The acquired data will then be stored in arrays or other variable storage memory locations which will be accessed by various functions.

3.5 Outputs Of The S-B CAD System

Figure 3.3 shows the various optional modes of the S-B CAD system output. After initialising and setting up of the graphic device, display background, colour, plotting border lines, axes, labels, and the generation of the user command menus; the program continuously scans the keyboard waiting for a selection to be made from the menu. The menu provides the following options:

1. stores 3-d data results representing real and imaginary of the closed-loop poles migration in the S-plane and their sensitivity magnitudes on X, Y and Z axes in the form a three column number of ASCII data file, or on a floppy disk for portability to other computer systems.

2. slow or fast display of the above results as 2-d or 3-d screen graphics, select angle of perspective or continuous graphic rotations, display boundary surface profile for parameter uncertainty analysis, superimpose plant fault path, adaptive controller decision path and control system operation path.

3. transfer the above data to another machine or machines through interface devices.

4. provide a hard copy of either the graphic results or the numerical values of all the data or selected data.

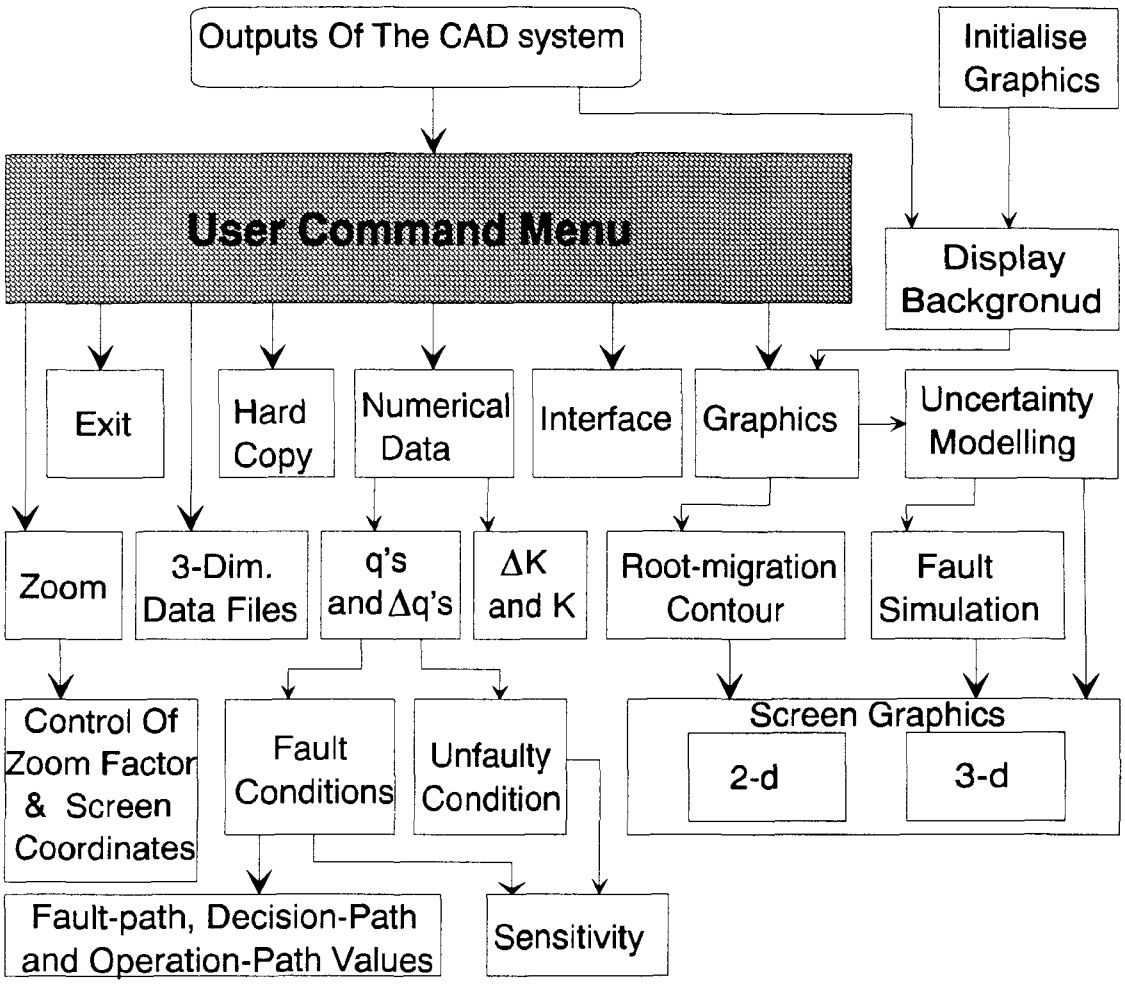


Fig. 3.3 The Various Optional Modes Of The S-B CAD System Output

3.5.1 The Projection Of 3-d Graphs Onto 2-d Screen

The 3-d co-ordinate point $^{q}{}_{d}$ (R_d, I_d, $|\mathbf{S}|_d$) representing the real & imaginary of the closed-loop poles migration in the S-plane and the magnitude of its sensitivity $|\mathbf{S}|_d$, respectively, can be projected onto a 2-d screen display (X_{screen}, Yp_{screen} and Y_{screen}) using the following transformation:

Referring to figure 3.4, (x, y, z) represent the projected 3-d axes onto the 2-d screen (X_{screen}, Y_{screen}), where x-axis is inclined by an angle "-θ " relative to x_{screen}, y-

axis is inclined by an angle "-ϕ" relative to Y_{screen} and z-axis coincides with Y_{screen}. Therefore, for a data point q_d, the transformed screen co-ordinates can be derived from the geometry as follows:

$$Xp_{screen} = R_d \cos\theta + I_d \sin\phi$$

$$Yp_{screen} = I_d \cos\phi - R_d \sin\theta$$

$$Y_{screen} = Yp_{screen} + z$$

$$= Yp_{screen} + |S|_d$$

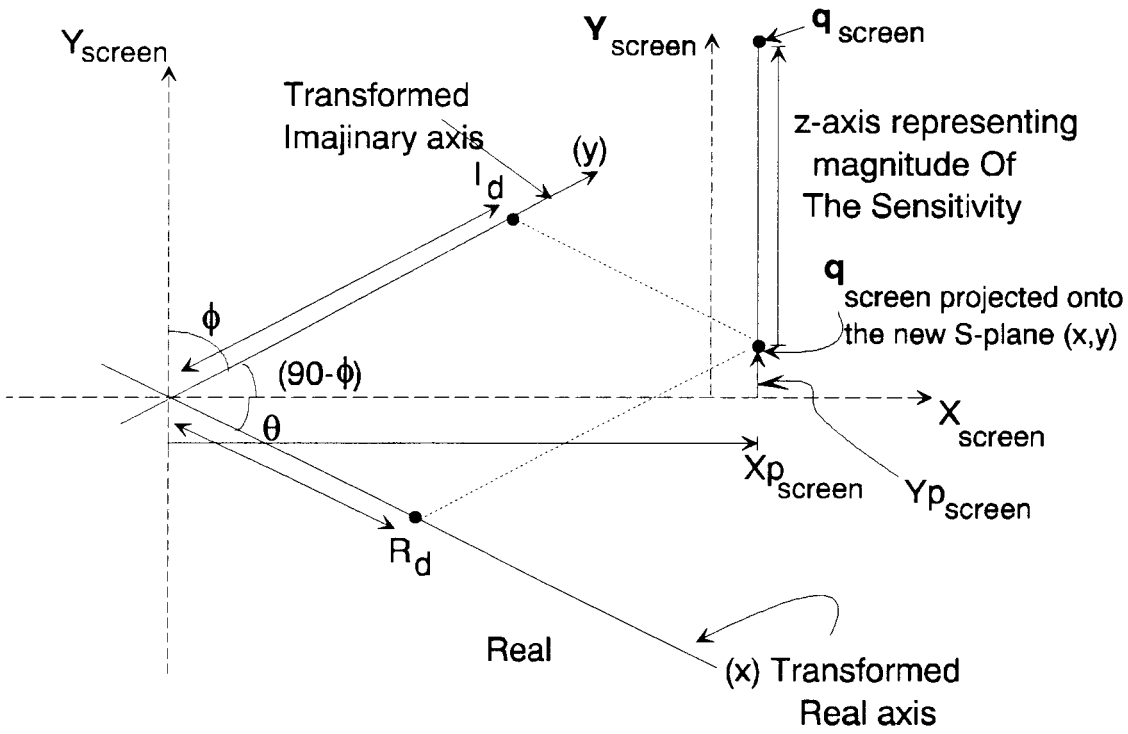


Figure 3.4 Details Of Projection Of 3-d Co-ordinate Data Point q_d (R_d, I_d, $|S|_d$) Onto 2-d Screen q_{screen} X_{screen}, Y_{screen}.

3.6 Testing The Physical Realisability

The program carries out a series of tests on the system's open-loop transfer function under consideration before activating the calculation cycles. This is to ensure that the system is physically realisable, i.e., first, the order of numerator polynomial is checked to be equal to or less than the denominator of the transfer function. Second, all the poles and zeros of the system with imaginary parts must be in pairs of complex conjugates i.e., equal real parts, as well as equal imaginary parts but with opposite signs. If the system is detected to be not physically realisable, the user is provided with options to re-enter the transfer function or to exit program execution.

3.7 Minimum Inter-Distance Calculations

The minimum inter-distance is defined as the smallest distance between any open-loop (then subsequently closed-loop) pole/zero pair. First all the squared inter-distances are calculated, as the sum of the squared difference of the real parts and the imaginary parts of every pole/zero pair as illustrated in figure 3.5. These values are then stored and sorted in the order of their magnitudes. The smallest value is finally selected and its square-root is calculated and stored for use in other functions of the program namely 'multiple pole' and 'dK cycle' routines. The flow diagram of this minimum inter-distance sub-routine is shown in Fig 3.6.

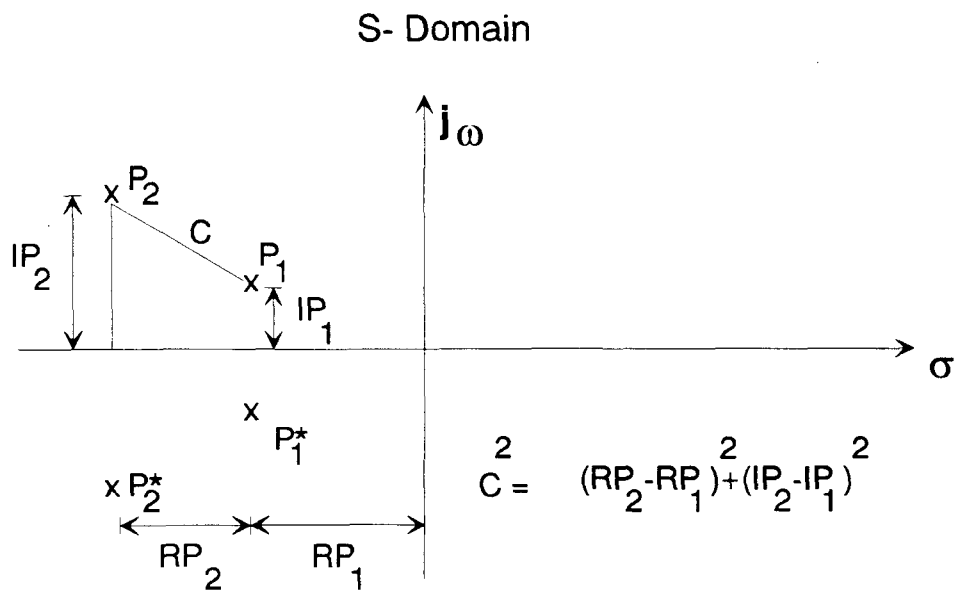


Figure 3.5 Details Illustrating The Determination Of The Squared Inter-Distance Between Every Pole/Zero Pair.

3.8 Detection Of High-Order Open-Loop Poles And Their Approximation By An Equivalent Cluster Of Distinct Poles

A high-order pole of order m is defined as m multiple poles located at the same spot in the S-plane. Floating point errors will occur when calculating the residues since, at exactly the high-order poles, a divide by zero would be encountered as pointed out earlier in section 2.3. The multiple pole subroutine will ensure that no more than one open loop-pole will be located at the same point in the S-plane, thereby eliminating this cause of floating point errors.

The first function of the subroutine is to detect the presence of two or more open-loop poles at a single location by measuring their inter-distances and comparing with zero. The poles detected at a single location are then grouped together and given a code for

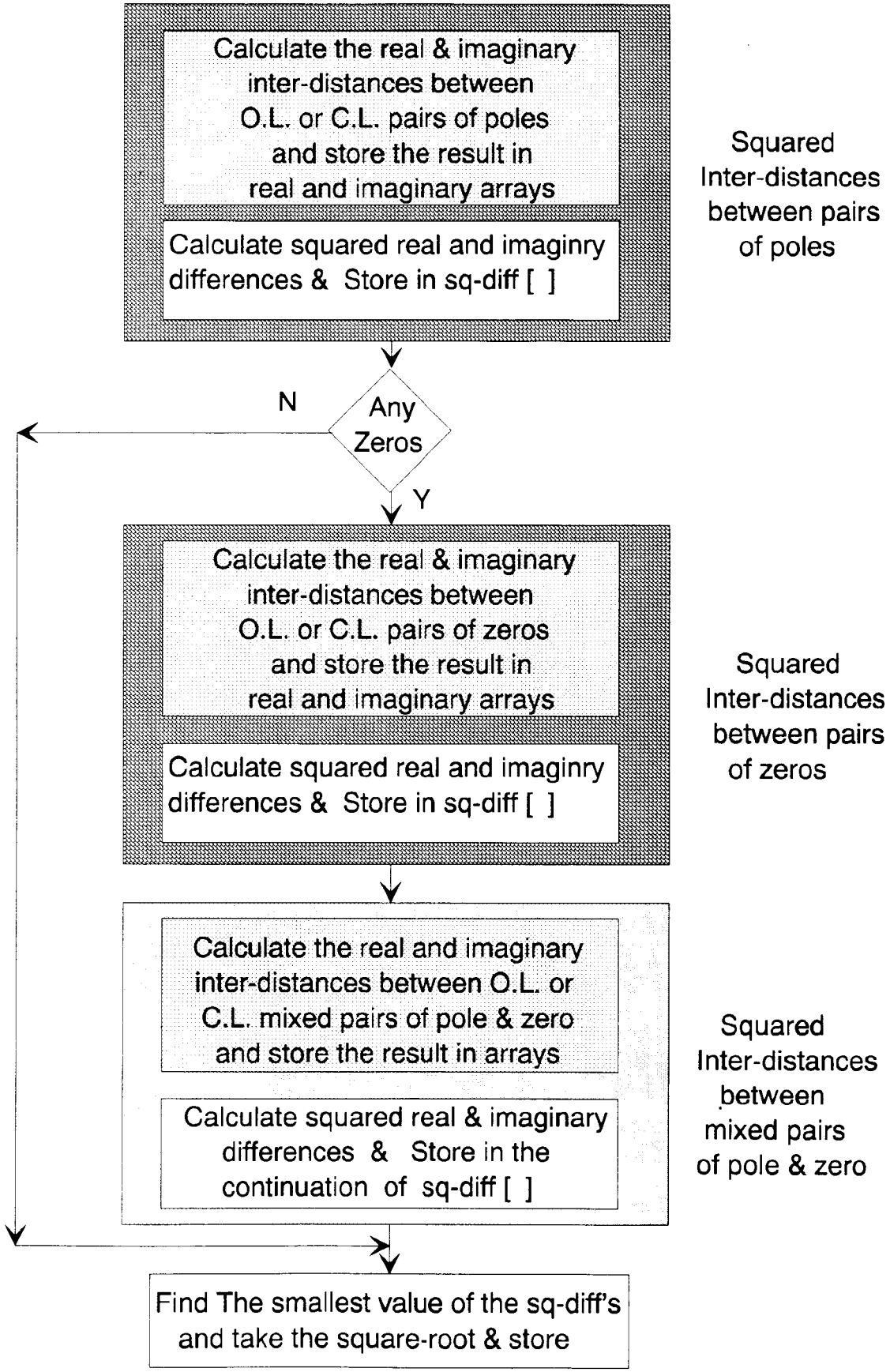


Figure 3.6 The Flow Diagram Of Minimum Inter-Distance Calculation Subroutine Between O.L. Or C.L. Pole/Zero Pairs.

future recall. Search among the rest of the poles then proceeds to detect other multiple poles upon which grouping and coding take place in a similar way.

Each group is then replaced by an equivalent cluster of distinct poles uniformly distributed in the vicinity of the high-order pole position as outlined in section 2.8.1. Figure 3.7 shows the flow diagram for the detection of high-order open-loop poles and the procedure of their replacement by equivalent cluster of distinct poles in their vicinity.

3.9 Cyclic Calculation Of The Loop-Gain Increment For Equi-Migration Steps Of Closed-Loop System Roots

At the beginning of each processing cycle, a subroutine is invoked to calculate the incremental step size of the loop-gain ΔK so as to induce, along any particular branch of the root contour, an equal migration step of the closed-loop poles. This has been already derived in section 2.8.2 as described by equation (63), reproduced here below as:

$$\Delta K_{(N)} = \frac{0.01\ epsi_a * \text{min-inter-distance between any pair of pole/zero of } T_{C.L.(N-1)}(S)}{|R_{i(N-1)max}|}$$

Where $epsi_a$ is a magnification/attenuation factor which has been introduced for the user convenience in case of over-riding the initial default value which is set as $epsi_a$ initial default $=1$. This initial default figure would be automatically reduced once the angle error-criterion had been violated as explained in section 2.8.2.

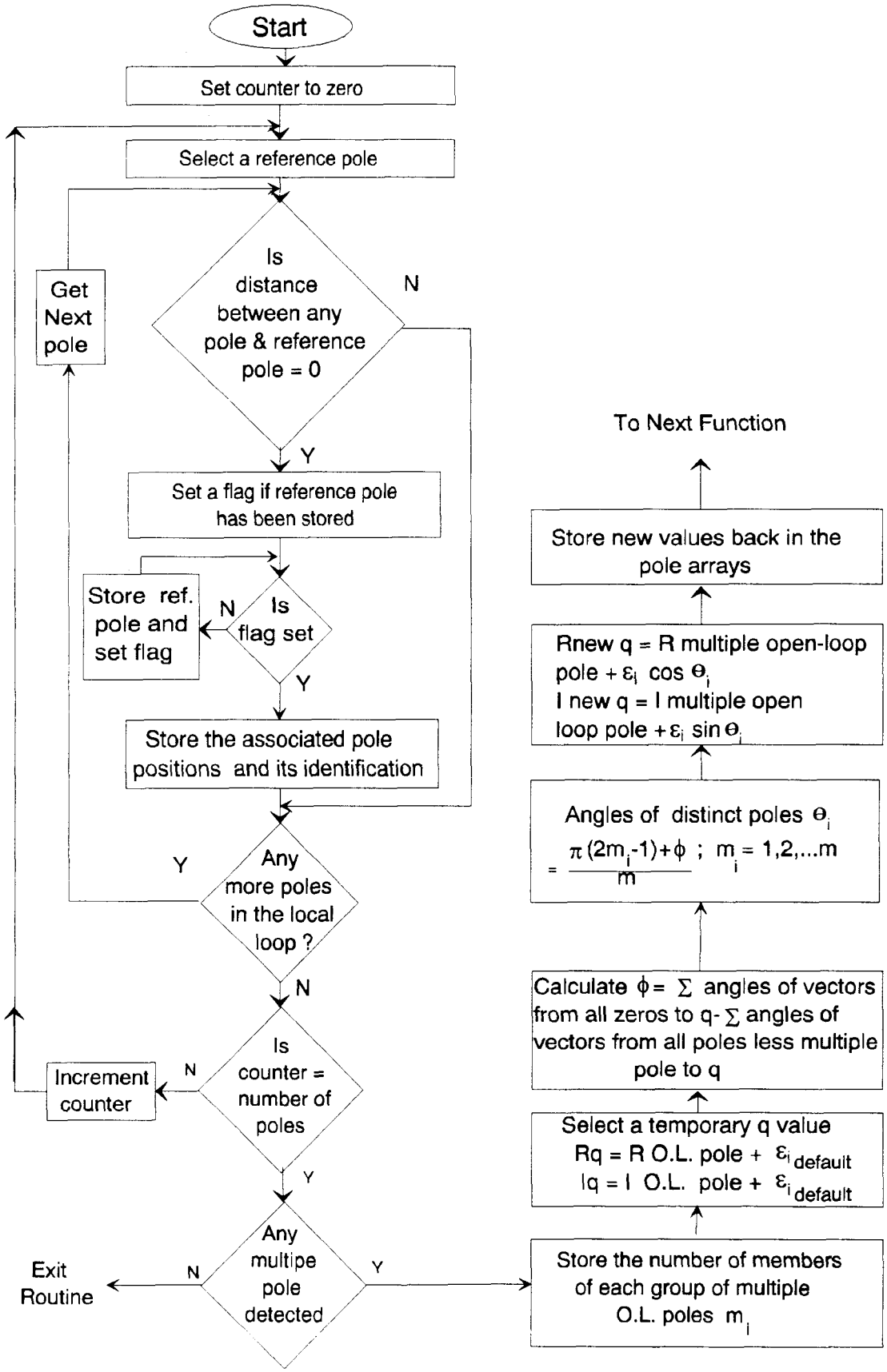


Figure 3.7 Flow Diagram Showing Method Of Detection Of High-Order Open -Loop Poles And Their Replacement By Equivalent Clusters Of Distinct Poles

3.10 Error Criterion For Maintaining A Prescribed Resolution

Theoretically the closed-loop poles migration contour should satisfy the angle criterion that is:

Net $\phi_{(N)}$ = Sum of the angles of vectors from all the updated open-loop zeros $Z_{j(N)}$ to the updated closed-loop pole $q_{i(N)}$ - sum of the angles of vectors from the updated open-loop poles $P_{j(N)}$ to the updated closed-loop pole $q_{i(N)} = \pm \pi$

where i refer to the ith branch of the migration contour and N refer to the computation cycle. There are, however, cumulating errors inherent in the estimation of the new positions $q_{i(N)}$ as explained in section 2.8.2, which depend principally on the step size of the loop gain ΔK, of the open-loop pole variations ΔP_j and of the open-loop zero variations ΔZ_j, as well as on the computer resolution. Therefore, the deviation $epsi_{(N)}$ in the estimated value of Net $\phi_{(N)}$ from the theoretical value of $\pm \pi$ should be less than or equal to a prescribed tolerable error as given by equation (62), otherwise ΔK, ΔP_j and ΔZ_j are reduced to their respective half values. This process should recursively continue until the above criterion is satisfied. The flow diagram which describe this functional subroutine is shown in figure 3.8.

During the implementation, it was recognised, however, that two types of ambiguities may arise in the calculation of Net $\phi_{(N)}$:

1- The component angles ϕ of Net $\phi_{(N)}$ are calculated individually as

arctan [Imaginary of $\{q_{i(N)} - P_{j(N)}\}$/ Real of $\{q_{i(N)} - P_{j(N)}\}$]

for vectors $[q_{i(N)} - P_{j(N)}]$ or as

arctan [Imaginary of $\{q_{i(N)} - Z_{j(N)}\}$/ Real of $\{q_{i(N)} - Z_{j(N)}\}$]

for vectors $[q_{i(N)} - Z_{j(N)}]$.

If the denominator is very small approaching zero, i.e., the angle of the particular vector is either $\pi/2$ or $3\pi/2$, floating point errors will occur causing breakdown to this function.

Otherwise, an unambiguous subroutine has to be devised to identify the quadrant of the argand plane within which each angle lies, i.e., either

0 to $\leq\pi/2$, $\pi/2$ to $\leq\pi$, π to $\leq 3\pi/2$, or $3\pi/2$ to $\leq/2\pi$.

These problems will be dealt with in the next section.

2- The calculated value of Net $\phi_{(N)}$, which is expected to be near $\pm\pi$, could be shifted by $\pm 2\pi$, or $\pm 4\pi$, ... in view of the nature of the equation with which it is determined. This ambiguity would cause a problem in the evaluation of the error; hence in the application of the error-criterion.

To eliminate this ambiguity, the following method was adopted:

* Find the integer multiple of 2π contained in $|\text{ Net } \phi_{(N)}|$; i.e.,

$$\text{Integer}\left(\frac{|\text{ Net } \phi_{(N)}|}{2\pi}\right)$$

* Therefore, $\text{epsi}_{(N)}$ of equation (62) can be deduced as follows:

$$|\text{epsi}_{(N)}| = \left| |\text{ Net } \phi_{(N)}| - 2\pi * \text{Integer}\left(\frac{|\text{ Net } \phi_{(N)}|}{2\pi}\right) \right|$$

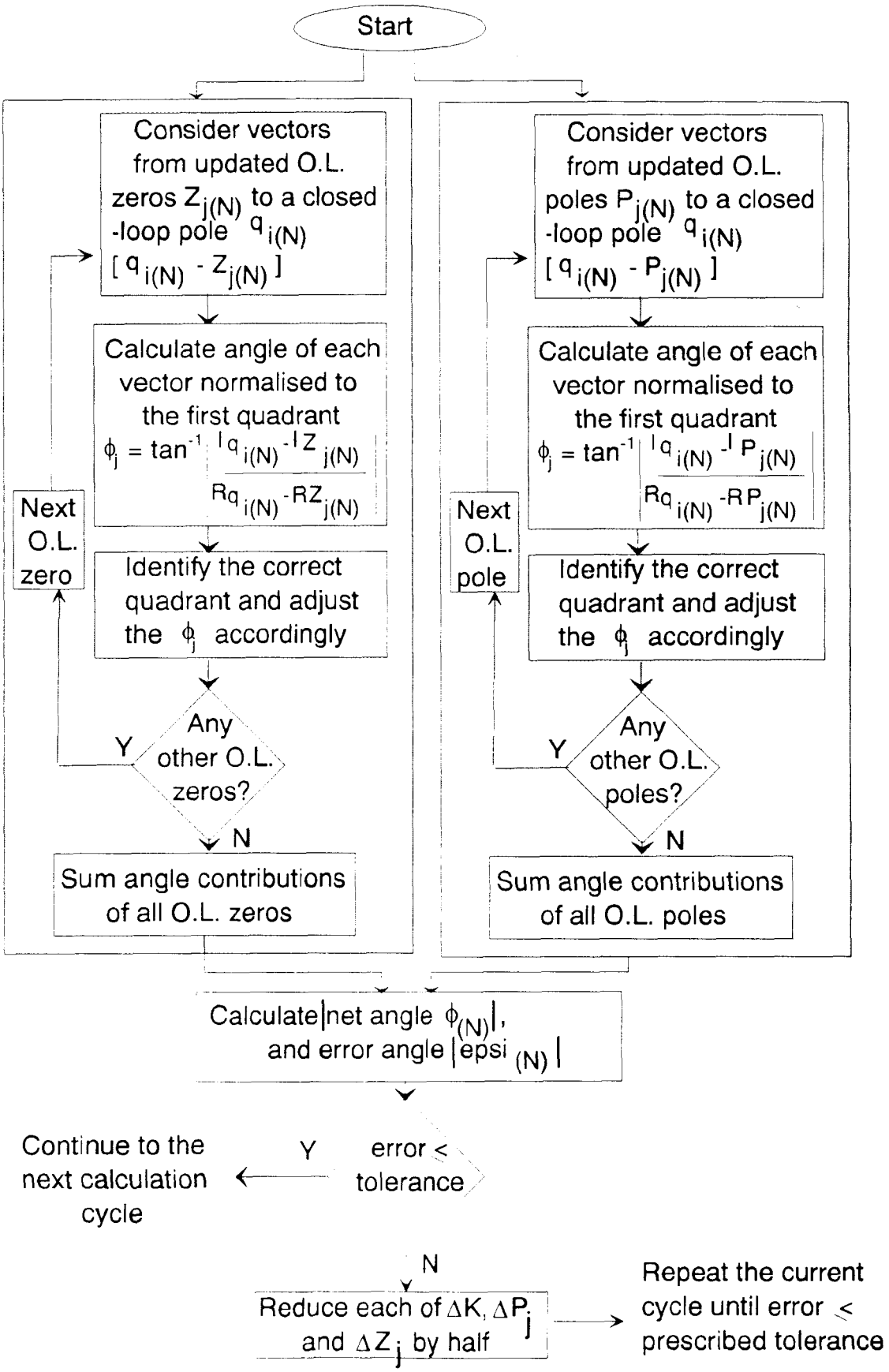


Figure 3.8 Flow Diagram Showing The Procedure For Maintaining A Prescribed Resolution Through The Application Of Angle Error-Criterion

3.11 Detection Of Saddle Points Along The Closed-Loop Poles Migration Contour And Computation Strategy In Their Vicinity Areas

At saddle points, the values of the residues; hence the root sensitivities would reach infinity, in theory, but computationally floating point errors followed with breakdown are expected. Therefore, a different computation strategy has been adopted in the vicinity of saddle points as outlined in section 2.8.3, provided these saddle points were detected before hand. Both the method of prediction of saddle points at the beginning of every computation cycle and the consequential strategy of calculations in the areas close to them have been software implemented with the help of the flow diagram shown in figure 3.9

To calculate the position of a saddle point, once it is predicted by the method described in section 2.8.3, the identified group of the closed-loop poles migrating towards it; $q_{i(N)}$; i= 1, 2, ...n, are assumed to lie uniformly on the circumference of a circle with radius ε_i (where $\varepsilon_i \longrightarrow o$) centred on the saddle point. Therefore the position of the saddle point can be calculated as the centre of gravity of this group of poles, i.e.,

$$\text{saddle point position } C = R\ \frac{\left(\sum_{i=1}^{i=n} q_{i(N)}\right)}{n},\ I\ \frac{\left(\sum_{i=1}^{i=n} q_{i(N)}\right)}{n},$$

where R and I refer to real and imaginary, respectively. Consequently, both the radius of the forbidden zone can then be computed as:

$$\varepsilon_i = \sqrt{\sum_{i=1}^{i=n} R^2[C-q_{i(N)}] + \sum_{i=1}^{i=n} I^2[C-q_{i(N)}]}\ ,$$

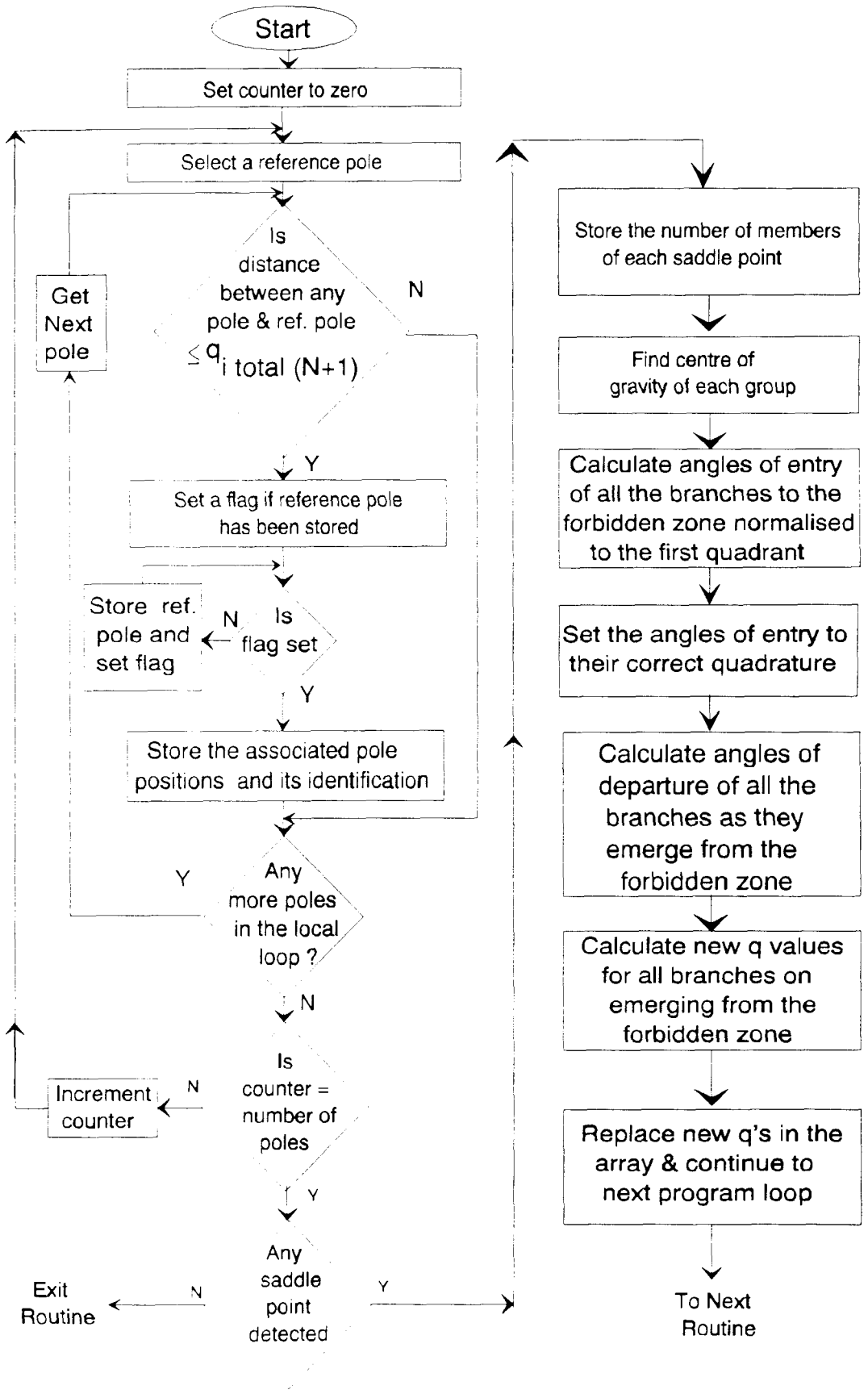


Figure 3.9 Showing Method Of Detection Of Saddle Points And The Implementation Of A Computation Strategy In Their Vicinity Areas.

and the angles of entry of the various branches of the migrating closed-loop poles $q_{i(N)}$ are:

θ_i = angle of the vector $[C-q_{i(N)})$; i=1,.....n

For instance $\theta_{i\ normalised}$ to the first quadrant of the argand plane is computed as:

$$\theta_{i\ normalised} = \arctan\left| \frac{I_C - Iq_{i(N)}}{R_C - Rq_{i(N)}} \right|$$

θ_i can then be determined once the correct quadrant is identified by the subroutine described in the next section.

The angles of departure" θ_{id} " can then be determined, as explained in section 2.8.3, as follows:

$$\theta_{id} = \theta_i + \frac{\pi}{n} \quad ; \ i=1, \ldots n$$

Therefore, the new positions of the closed-loop poles $q_{i(N+1)}$ at the point of emerging from the forbidden zone can be computed as follows:

$$q_{i(N+1)} = C + R\,[\varepsilon_i \cos\theta_{id}] + I\,[\varepsilon_i \sin\theta_{id}] \quad ; \ i=1, \ldots n$$

and the corresponding loop-gain $K_{(N+1)}$ can be computed using equation (71) of section 2.8.3.

3.11.1 Elimination Of Ambiguities In Angle Calculations

In section 3.10 and 3.11 calculation of the angle of inclination of vectors in the S-plane are required particularly in connection with the application of an error criterion for a prescribed computation resolution throughout, as well as in connection with the critical computation in the vicinity of saddle points. Two types of ambiguities may arise, as pointed out in section 3.10, which need to be resolved.

Before proposing the solutions, it is appropriate to start with defining the nature of the problems as follows:

for a vector between two points (x_1, y_1) and(x_2, y_2) in the S-plane, which may represent the locations of poles or zeros, Δx and Δy may be defined as follows:

$\Delta x = x_2 - x_1$

$\Delta y = y_2 - y_1$

The angle of inclination of this vector " θ " can be calculated as:

$$\theta = \arctan \frac{\Delta y}{\Delta x}$$

The first problem arises when Δx approaches zero since $\frac{\Delta y}{\Delta x}$ will approach infinity causing floating point errors and eventual breakdown of this subroutine. To overcome this problem, it is necessary to predict the problem before proceeding or otherwise with the computation of the ratio $\frac{\Delta y}{\Delta x}$. This can be achieved by comparing the value of $| \Delta x |$ with a parameter epsi having a very small value as to maintain an overall prescribed resolution i.e., the value of epsi must be related to $epsi_{(N)}$ (as defined by equation (62) in section 2.8.3) as a small fraction of it. Therefore, a default value for epsi has been selected as:

$$epsi = 0.00001 \frac{\left(\Delta K_{(N)} + \Delta P_{j(N)min} + \Delta Z_{j(N)min} \right)}{1 + \text{No. of varing O.L. poles } n_1 + \text{No. of varying O.L. zeros } n_2}$$

therefore,

If $| \Delta x | \leq epsi$, $y_2 > y_1$ (or $\Delta y > 0$), then the value of θ is returned as $\pi/2$ radians,

if $| \Delta x | \leq epsi$, $y_2 < y_1$ (or $\Delta y < 0$), then the value of θ is returned as $3\pi/2$ radians,

otherwise (i.e., when $| \Delta x | > epsi$), $\left| \frac{\Delta y}{\Delta x} \right|$ is calculated such that

θ normalised to the first quadrant $= \arctan \left| \frac{\Delta y}{\Delta x} \right|$ of the argand plane

In the last case, the correct quadrant of θ must identified in order to determine the actual value of θ such that $0 \leq \theta \leq 2\pi$ excluding a narrow zone around $\pi/2$ and $3\pi/2$ which have been catered for in the previous two cases. This procedure is described below:

θ is lying in the first quadrant of the argand plane (0 to $\pi/2$)

If $\Delta y \geq 0$ and $\Delta x > 0$ then

$\theta_{actual} = \theta_{normalised}$

θ is lying in the second quadrant of the argand plane ($\pi/2$ to π)

If $\Delta y \geq 0$ and $\Delta x < 0$ then

$\theta_{actual} = \pi - \theta_{normalised}$

θ is lying in the third quadrant of the argand plane (π to $2\pi/3$)

If $\Delta y \leq 0$ and $\Delta x < 0$ then

$\theta_{actual} = \pi + \theta_{normalised}$

θ is lying in the third quadrant of the argand plane ($2\pi/3$ to 2π)

If $\Delta y \leq 0$ and $\Delta x > 0$ then

$\theta_{actual} = 2\pi - \theta_{normalised}$

3.12 Computation Of The Closed-Loop Poles' Sensitivities And Updation Of Their Locations Along Their Migration Contour

In chapter 2, it has been shown that the closed-loop pole sensitivity can be computed for a particular loop gain K using any single or combinations of equations (41), (43) and (45), provided that the normalised residue R_i is known as a priori. However, at the end of each computation cycle the up-dated positions of the closed-loop poles are available, thereby the respective normalised residues as well as the associated sensitivities can be computed using equations (56) and (67) respectively. For K=0, however, R_i & $S_K^{q_i}$ are computed by equations (51) and (52) instead as explained in detail in section 2.8.1, after replacing any high-order open-loop poles by their equivalent clusters of distinct poles.

To update the locations of the closed-loop poles along their migration trajectories, estimation of their respective total incremental migrations steps during a computation cycle has to be carried out using equation (66), provided that the new locations are not saddle points, followed by computation of the recursive equation (60). If some or all the new Locations are saddle points, then the locations of the closed-loop poles as they emerge from the forbidden zone are computed using the procedure described in section 3.11.

For the former case, the resolution-enhanced scheme outlined in section 2.8.3 is adopted, where each computation cycle is divided into three substeps. These have been software implemented using the flow diagram shown in figure 3.10.

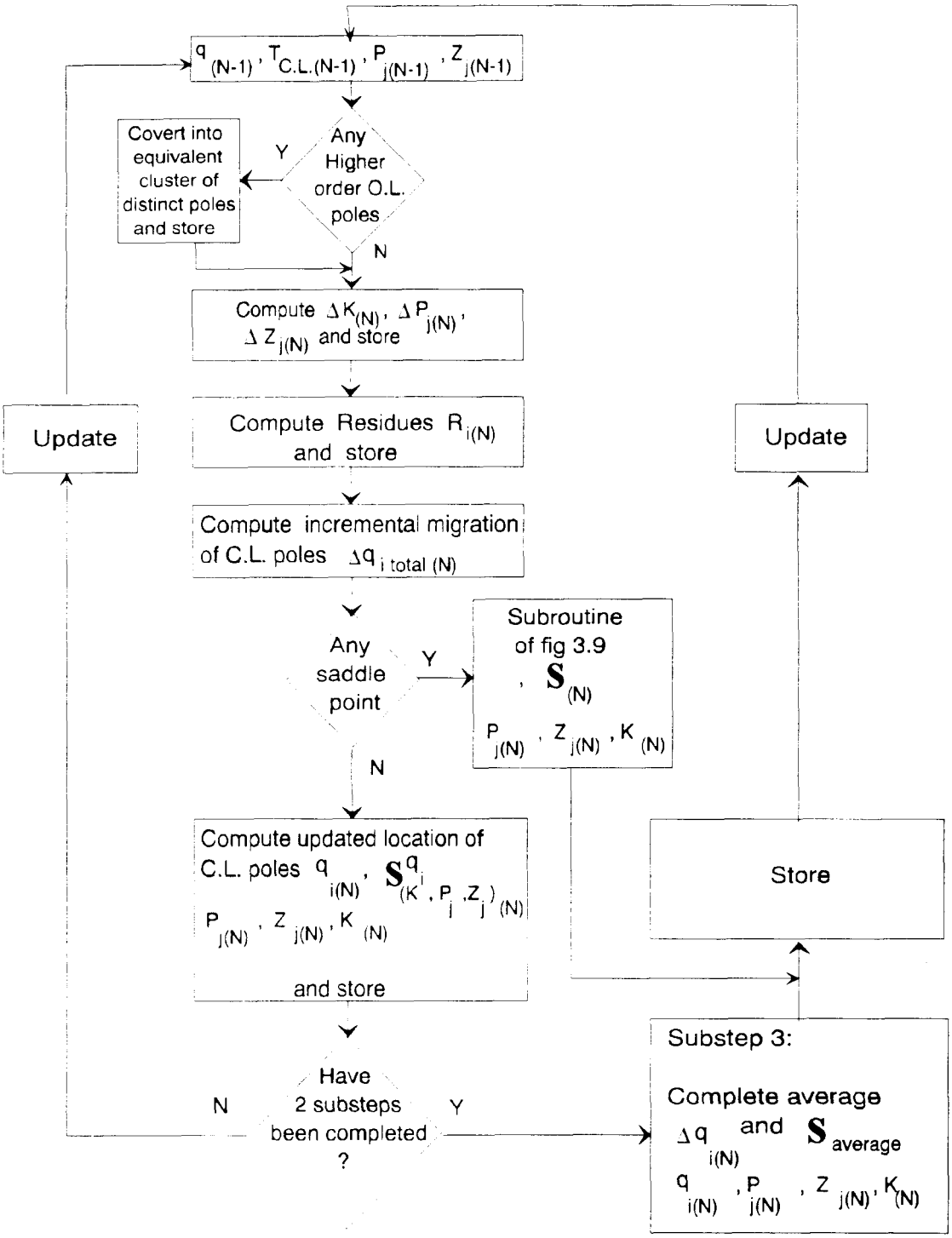


Figure 3.10 Flow Diagram Showing A Resolution-Enhanced Three Substep Computation Cycle Of The Closed-Loop Poles' Sensitivities And Their New Locations Along Their Migration Contour Outside The Forbidden Zone Surrounding The Saddle Points.

3.13 Fault Simulation And Modelling Of Parameter Uncertainty Of Plants

In chapter 6, parameter uncertainty or perturbations are shown to cause unpredictable migration of the open-loop poles and/or zeros within relatively small fuzzy zones in the S-plane surrounding their nominal positions. In a somewhat similar way, plants experiencing a common class of fault, manifested in the form of gradual degradation of the characteristics, can also be interpreted in terms of migrations of some or all of the open-loop poles and/or zeros, with different speeds generally, within fault-regions as outlined in chapter 7. Even in the absence of precise knowledge of the paths of migration of the poles/zeros within their respective fuzzy zones or the faulty regions, it is nevertheless possible to encompass all these likely paths by considering migration along the boundaries of these regions. Having defined the boundaries of these regions which represent the worst possible condition (through the selection of the sizes and shapes of these regions), the incremental pole/zero migration ΔP_j and ΔZ_j (refer to equation (64) and (65) of section 2.8.2) and the number of steps can be determined such that the computation resolution can be maintained within a prescribed tolerance as required for a specific application. The relative values of ΔP_j's and ΔZ_j's will then reflect the relative speeds of migration to represent a wide diversity of parameter perturbations or fault conditions.

The effects of ΔP_j's and ΔZ_j's on the prediction of the overall migration contour of the closed-loop poles and their sensitivities have been incorporated in each computation cycle as outlined in the previous two sections. It follows, therefore, that in the absence of fault and parameter uncertainty, the values of ΔP_j's and ΔZ_j's should be set to zero.

A wide range of scenarios of fault simulation and parameter uncertainty can be examined by the user through manipulation of ΔK, ΔP_j's and ΔZ_j's. For instance, ΔP_j's and ΔZ_j's may be set to zero while ΔK is left finite to allow the examination of the

system's behaviour under unfaulty deterministic conditions until K reaches a certain operating value K_1, when either fault or uncertainty is assumed to occur by allowing ΔP_j's and ΔZ_j's to assume finite values. In this way, the effects of either faults and/or parameter perturbations on the system's composite performance-sensitivity can be highlighted near the normal operating region.

3.14 Program Coding And Development Environment

The coding of the CAD system was carried out in Turbo C* [58] (Appendix A gives the complete program listing of the S-B CAD system). The high-level language Turbo C language was chosen because of its portability from one system to another; being structured, modular, compiled. The editor of Turbo C operating system was deployed in the development, compilation, execution, testing and expansion of the program on an IBM compatible PC.

3.15 Summary

The work in this chapter has been primarily targeted to the software implementation of the S-B algorithm as a highly flexible CAD system suitable for analysis and design of robust, fault-tolerant control systems having plants subject to all types of uncertainty and/or undergoing a diverse range of faults belonging to a common class, and operate in a real noisy environment, as will be discussed in chapters 6 & 7.

---

* Turbo C supports the draft-proposed American National Standards Institute (ANSI) C standard, fully supports the Keringham and Ritchie definition, and includes certain optional extensions for mixed-language and mixed model programming.

Emphasis has been placed on accuracy, automated procedures to diagnose and cater for special situations, elimination of ambiguities, self evaluation and on integration of an adaptive graphic facilities together with a helpful range of user-machine interactions is explained by a collection of flow diagrams. The development, testing and debugging have been carried out in the Turbo C editor environment which has the distinctive advantages of flexibility and portability.

The potential of the S-B CAD system application will be explored in chapters 5, 6 and 7, after conducting a comprehensive evaluation of all aspects of its performance objectives and quality assurance in chapter 4.

Chapter 4

Composite Sensitivity- Performance Knowledge-Base Classification Of Feed-Back Systems With Deterministic Plants

4.1 Introduction

After completion of the software implementation of the sensitivity-based algorithm and of the preliminary debugging of the various subroutines, it was necessary to conduct an extensive experimental programme to evaluate all aspects of its performance objectives including accuracy, speed and its quality assurance. The programme was divided into two stages:

Stage 1: The algorithm was activated to estimate and plot the root-loci and the root-sensitivity diagrams of a large number of first-order and second-order feedback system models having diverse open-loop pole/zero deterministic plant configurations and delays. The results of both the data and the graphics were then subjected to rigorous comparisons with the theoretical solutions, which are readily available, in terms of accurately predictable geometric shapes, locations of breakaway points, locations of intersection points with the real-axis and/or the imaginary axis of the S-plane and the values of the corresponding loop gains and sensitivities.

Stage 2: Stage 1 of the experimental programme was repeated using extensive examples of higher order feedback systems featuring a wide range of deterministic plant configurations and delays. The results of both the S-domain and the frequency-

domain loci were then contrasted with those obtained from the commercially available CAD system: Program CC version 4.0, see reference [57].

As a result of these tests, the superiority of the sensitivity-based algorithm over its counterpart became evident in its ability to produce directly, and almost in real-time, composite 3-d performance-sensitivity graphics for systems with greater accuracy. These advantages were exploited in generating a graphic knowledge-base classification of feed-back systems incorporating deterministic plants of diverse configurations and delays, as described in section 4.3.

The accessibility to this vast multi-aspect information together with the inbuilt manipulating power and speed of the screen graphic display, in either interactive or automatic modes, indicate the potential of this knowledge-base as a valuable CAD tool as will be revealed in chapters 5, 6 & 7.

4.2 Performance Evaluation Of The Sensitivity-Based Algorithm

Contrasting the results obtained from stage 1 of the testing programme, referred to in the previous section, with those obtained theoretically, it was found that the algorithm consistently produced the correct composite graphics with a resolution limited only by the default values of the angle error-criterion and of the step size (refer to chapter 3), in addition to the computation resolution. The resolution of the algorithm, however, was found to be so high- almost comparable to the computer resolution- to conclude that the errors due to the inbuilt defaults are insignificant.

Conducting stage 2 of the testing program was far more difficult since the sensitivity information were not readily available from the commercial CAD system. This were

eventually obtained indirectly through complicated manipulations and the use of a mathematics CAD system (see reference [72]). Furthermore, it was discovered that the commercial CAD system uses an approximation method (all-pass polynomial) for modelling plant delays; resulting in considerable errors particularly as the calculation zone moves away from the imaginary axis and deeper into the S-plane. However, by using the theoretically derived results as standard bench-marks, it was possible to carry out comparative studies with deductions extrapolated for tests with higher order systems.

The results of these tests revealed that the developed S-B CAD system is far superior over its commercial counterpart in terms of accuracy, speed, range of automated facilities, accessibility to a vast array of the modelled system behaviour either in detailed step-by-step viewing mode or in continuously updating mode.

4.3 Knowledge-Base Sensitivity Classification(KBSC)

The graphic knowledge-base is given in Appendix B of this thesis. This has been organised as follows:

4.3.1 First-Order Plants

The group of figures 4.1A to 4.1E show the S-plane root- locus diagram (RL) of first-order plants classified according to the open - loop pole /zero configuration. The sensitivity profile(SP) of the closed-loop poles for each plant is displayed on a series of companion 3-d graphs, classified according to the amount of the associated pure delay. The open-loop pole/zero configurations are specified as follows:

*A deterministic plant with a transfer function, $G(S) = \frac{K}{S}$, (RL) and (SP) are illustrated in figure 4.1A, the open-loop pole is located at the origin and the open-loop-zero at infinity, of the S-plane. Figure 4.1a shows the relevant (SP) graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K\,(S+2)}{(S+1)}$, (RL) and (SP) are illustrated in figure 4.1B, the open-loop pole is located at -1, and the open-loop-zero at -2, of the S-plane. Figure 4.1b shows the relevant (SP) graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K\,(S+1)}{(S+2)}$, (RL) and (SP) are illustrated in figure 4.1C, the open-loop pole is located at -2, and the open-loop zero at -1, of the S-plane. Figure 4.1c shows the relevant (SP) graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K\,S}{(S+2)}$, (RL) and (SP) are illustrated in figure 4.1D, the open-loop pole is located at -2, and the open-loop zero at the origin, of the S-plane. Figure 4.1d shows the relevant (SP) graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K\,(S+2)}{(S+2)}$, (RL) and (SP) are illustrated in figure 4.1E, the open-loop pole is located at -2, and the open-loop-zero at -2, of the S-plane. Figure 4.1e shows the relevant (SP) graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

Examining the above group of 3-d graphs should reveal the functional dependence of the sensitivity profile of the close-loop pole on both the open-loop pole/zero pattern

and the pure delay. It is interesting to note that, throughout, the general shape of the root- locus diagram of each plant remains invariant to the amount of pure delays.

Combined (RL) And (SP) Graphs For A set Of First-Order plant Configurations*

Since the graphic knowledge-base contains vast information, it would be substantially easier, for comparative study, to combine the relevant sensitivity profiles into one or more family set of 3-d diagrams. The set of the diagrams is then rotated around the sensitivity axis to give the best unambiguous viewing angle of perspective. For instance, figure 4.1 illustrates the family of sensitivity versus the real axis of the S-plane of the following set of first-order family plants' combinations with zero delay:

$$G_1(S) = \frac{K}{(S+1)}, G_2(S) = \frac{K(S+1.5)}{(S+1)}, G_3(S) = \frac{K(S+0.5)}{(S+1)}, G_4(S) = \frac{K(S+2)}{(S+1)}, G_5(S) = \frac{K S}{(S+1)}, G_6(S) = \frac{K(S+5)}{(S+1)}, G_7(S) = \frac{K(S-3)}{(S+1)}, G_8(S) = \frac{K(S+6)}{(S+1)}, G_9(S) = \frac{K(S+8)}{(S+1)}.$$

4.3.2 Second-Order Plants

Unlike the first-order plants, the second-order plants produces a substantially increased diversity of open-loop pole/zero configurations spanning, theoretically, the entire space of the S-plane, instead of being confined to the real-axis. Furthermore, the root- locus diagram of each plant is no longer invariant to the amount of pure delay. Therefore, the graphic knowledge base classification would have to reflect these extra degrees of freedom.

* The combined plants represent examples some of which are not included in the concise knowledge-base given in appendix B.

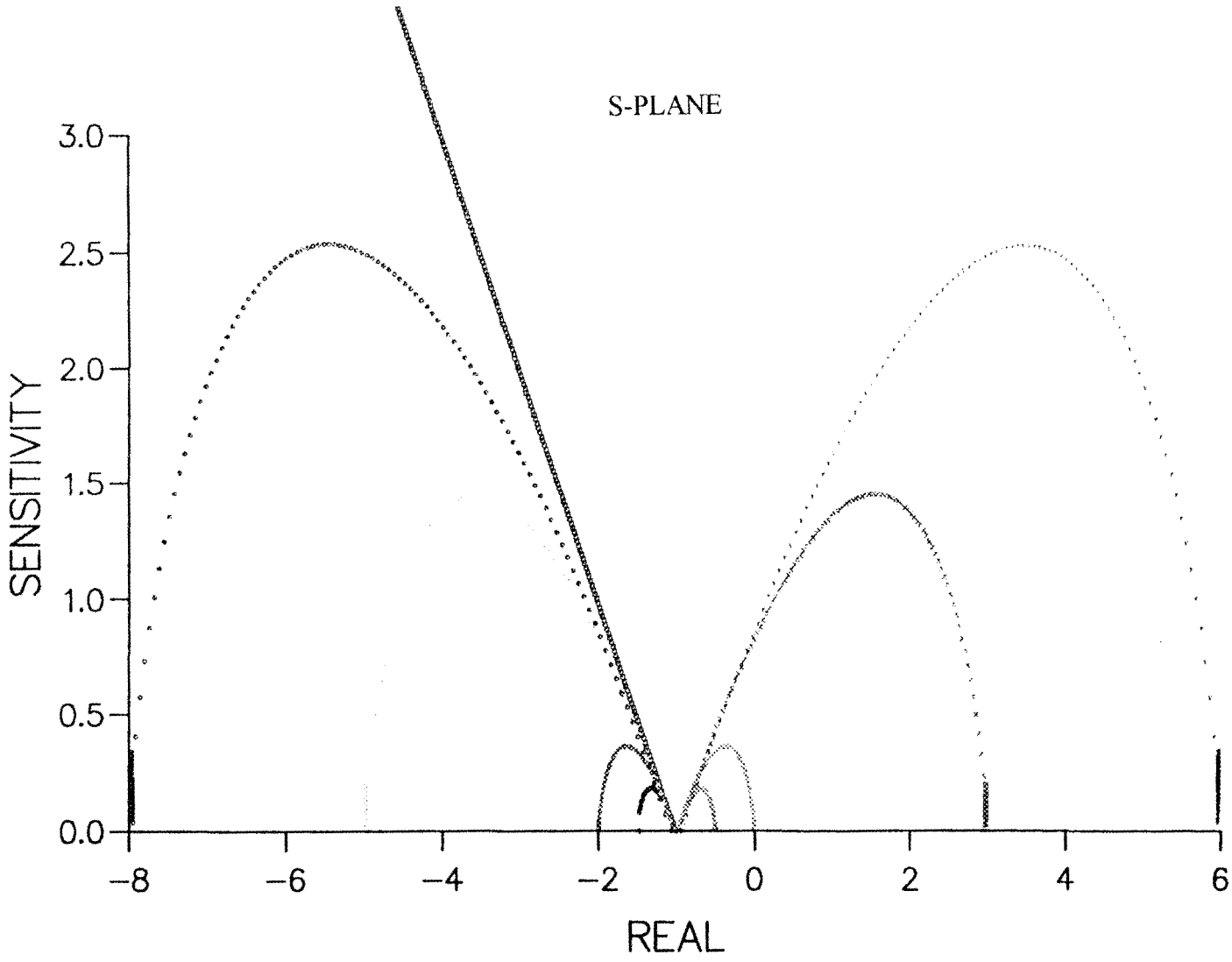


FIGURE 4.1 COMBINED FIRST-ORDER PLANTS' SENSITIVITY PROFILES

The group of figures 4.2 A to 4.2 #N shows the S-plane root- locus diagrams (RL) of second- order plants accompanied by the root-sensitivity profile (SP) classified according to the open - loop pole /zero configuration. For each configuration, the relevant graphs have been captured and classified according to the amount of pure delay associated with the plant. The open-loop pole/zero configurations are specified as follows:

*A deterministic plant with a transfer function, $G(S) = \frac{K}{(S^2+4S+3)}$, (RL) and (SP) are illustrated in figure 4.2A, the open-loop poles are located at -1 and at -3, and two open-loop zeros at infinity, of the S-plane. Figure 4.2a shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S+4)}{(S^2+4S+3)}$, (RL) and (SP) are illustrated in figure 4.2B, the open-loop poles are located at -1 and at -3, and the open-loop zeros at infinity and at -4, of the S-plane. Figure 4.2b shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S+3)}{(S^2+5S+4)}$, (RL) and (SP) are illustrated in figure 4.2C, the open-loop poles are located at -1 and at -4, and the open-loop zeros at infinity and at -3, of the S-plane. Figure 4.2c shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S+1)}{(S^2+6S+8)}$, (RL) and (SP) are illustrated in figure 4.2D, the open-loop poles are located at -2 and at -4, and the open-loop zeros at infinity and at -1, of the S-plane. Figure 4.2d shows the

relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{KS}{(S^2+4S+3)}$, (RL) and (SP) are illustrated in figure 4.2E, the open-loop poles are located at -1 and at -3, and the open-loop zeros at infinity and at the origin , of the S-plane. Figure 4.2e shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S+1)}{(S^2+4S+3)}$, (RL) and (SP) are illustrated in figure 4.2F, the open-loop poles are located at -1 and at -3, and the open-loop zeros at infinity and at-1 , of the S-plane. Figure 4.2f shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+9S+20)}{(S^2+4S+3)}$, (RL) and (SP) are illustrated in figure 4.2G, the open-loop poles are located at -1 and at -3, and the open-loop zeros at -4 and at -5 , of the S-plane. Figure 4.2g shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+6S+8)}{(S^2+4S+3)}$, (RL) and (SP) are illustrated in figure 4.2H, the open-loop poles are located at -1 and at -3, the open-loop zeros at -4 and at -2 , of the S-plane. Figure 4.2h shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+5S+6)}{(S^2+5S+4)}$, (RL) and (SP) are illustrated in figure 4.2I, the open-loop poles are located at -1 and at -4, the

open-loop zeros at -2 and at -3 , of the S-plane. Figure 4.2i shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+4S+3)}{(S^2+6S+8)}$, (RL) and (SP) are illustrated in figure 4.2J, the open-loop poles are located at -2 and at -4, and the open-loop zeros at -1 and at -3 , of the S-plane. Figure 4.2j shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+3S+2)}{(S^2+7S+12)}$, (RL) and (SP) are illustrated in figure 4.2K, the open-loop poles are located at -3 and at -4, and the open-loop zeros at -1 and at -2 , of the S-plane. Figure 4.2k shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2-3S+2)}{(S^2+3S+2)}$, (RL) and (SP) are illustrated in figure 4.2L, the open-loop poles are located at -1 and at -2, and the open-loop zeros at +1 and at +2 , of the S-plane. Figure 4.2l shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K}{(S^2+4S+8)}$, (RL) and (SP) are illustrated in figure 4.2M, the complex conjugate pair of open-loop poles are located at $-2 \pm j2$, and two open-loop zeros at infinity, of the S-plane. Figure 4.2m shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds

*A deterministic plant with a transfer function, $G(S) = \frac{K(S+4)}{(S^2+4S+8)}$, (RL) and (SP) are illustrated in figure 4.2N, the complex conjugate pair of open-loop poles are located at $-2 \pm j2$, and the open-loop zeros at infinity and at -4, of the S-plane.

Figure 4.2n shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S+2)}{(S^2+4S+8)}$, (RL) and (SP) are illustrated in figure 4.2 O, the complex conjugate pair of open-loop poles are located at $-2 \pm j2$, and the open-loop zeros at infinity and at -2, of the S-plane. Figure 4.2 shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K S}{(S^2+4S+8)}$, (RL) and (SP) are illustrated in figure 4.2P, and the complex conjugate pair of open-loop poles are located at $-2 \pm j2$, and the open-loop zeros at infinity and at the origin, of the S-plane. Figure 4.2p shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S-1)}{(S^2+4S+8)}$, (RL) and (SP) are illustrated in figure 4.2Q, the complex conjugate pair of open-loop poles are located at $-2 \pm j2$, and the open-loop zeros at infinity and at +1, of the S-plane. Figure 4.2q shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+7S+12)}{(S^2+4S+8)}$, (RL) and (SP) are illustrated in figure 4.2R, the complex conjugate pair of open-loop poles are located at $-2 \pm j2$, and the open-loop zeros at -4 and at -3, of the S-plane. Figure 4.2r shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+5S+6)}{(S^2+4S+8)}$, (RL) and (SP) are illustrated in figure 4.2S, the complex conjugate pair of open-loop poles are located at -2 $\pm$ j2 , and the open-loop zeros at -2 and at -3, of the S-plane. Figure 4.2s shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+3S+2)}{(S^2+4S+8)}$, (RL) and (SP) are illustrated in figure 4.2T, the complex conjugate pair of open-loop poles are located at -2 $\pm$ j2 , and the open-loop zeros at -2 and at -1, of the S-plane. Figure 4.2t shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{KS(S+1)}{(S^2+4S+8)}$, (RL) and (SP) are illustrated in figure 4.2U, the complex conjugate pair of open-loop poles are located at -2 $\pm$ j2 , and the open-loop zeros at -1 and at the origin, of the S-plane. Figure 4.2u shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2-3S+2)}{(S^2+4S+8)}$, (RL) and (SP) are illustrated in figure 4.2V, the complex conjugate pair of open-loop poles are located at -2 $\pm$ j2 , and the open-loop zeros at +1 and at +2, of the S-plane. Figure 4.2v shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+8S+17)}{(S^2+4S+8)}$, (RL) and (SP) are illustrated in figure 4.2W, the complex conjugate pair of open-loop poles are located at -2 $\pm$ j2 , and the complex conjugate pair of open-loop zeros at -4 $\pm$ j1, of the

S-plane. Figure 4.2w shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+4S+5)}{(S^2+4S+8)}$, (RL) and (SP) are illustrated in figure 4.2X, the complex conjugate pair of open-loop poles are located at $-2 \pm j2$, and the complex conjugate pair of open-loop zeros at $-2 \pm j1$, of the S-plane. Figure 4.2x shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+2S+2)}{(S^2+4S+8)}$, (RL) and (SP) are illustrated in figure 4.2Y, the complex conjugate pair of open-loop poles are located at $-2 \pm j2$, and the complex conjugate pair of open-loop zeros at $-1 \pm j1$, of the S-plane. Figure 4.2y shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+1)}{(S^2+4S+8)}$, (RL) and (SP) are illustrated in figure 4.2Z, the complex conjugate pair of open-loop poles are located at $-2 \pm j2$, and the complex conjugate pair of open-loop zeros at $0 \pm j1$, of the S-plane. Figure 4.2z shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+1)}{(S^2+4S+20)}$, (RL) and (SP) are illustrated in figure 4.2#A, the complex conjugate pair of open-loop poles are located at $-2 \pm j4$, and the complex conjugate pair of open-loop zeros at $0 \pm j1$, of the S-plane. Figure 4.2#a shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+1)}{(S+2)^2}$, (RL) and (SP) are illustrated in figure 4.2#B, the multiple open-loop poles are located at -2 , and the complex conjugate pair of open-loop zeros at $0 \pm j1$, of the S-plane. Figure 4.2#b shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2-2S+2)}{(S^2+4S+8)}$, (RL) and (SP) are illustrated in figure 4.2#C, the complex conjugate pair of open-loop poles are located at $-2 \pm j2$, and the complex conjugate pair of open-loop zeros at $+1 \pm j1$, of the S-plane. Figure 4.2#c shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K}{S^2}$, (RL) and (SP) are illustrated in figure 4.2#D, the multiple open-loop poles are located at the origin, and two open-loop zeros at infinity, of the S-plane. Figure 4.2#d shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S+2)}{S^2}$, (RL) and (SP) are illustrated in figure 4.2#E, the multiple open-loop poles are located at the origin, and the open-loop zeros at -2 and at infinity, of the S-plane. Figure 4.2#e shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{KS}{(S^2+4S+4)}$, (RL) and (SP) are illustrated in figure 4.2#F, the multiple open-loop poles are located at -2, and the open-loop zeros at the origin and at infinity, of the S-plane. Figure 4.2#f shows

the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+6S+9)}{(S^2+2S+1)}$, (RL) and (SP) are illustrated in figure 4.2#G, the multiple open-loop poles are located at -1, and the multiple open-loop zeros at -3, of the S-plane. Figure 4.2#g shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{KS^2}{(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.2#H, the multiple open-loop poles are located at -3, and the multiple open-loop zeros at the origin, of the S-plane. Figure 4.2#h shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2-4S+4)}{(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.2#I, the multiple open-loop poles are located at -3, and the multiple open-loop zeros at +2, of the S-plane. Figure 4.2# i shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2-2S+2)}{(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.2#J, the multiple open-loop poles are located at -3, and the complex conjugate pair of open-loop zeros at $+1 \pm j1$, of the S-plane. Figure 4.2# j shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+1)}{(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.2#K, the multiple open-loop poles are located at -3, and

the complex conjugate pair of open-loop zeros at $0 \pm j1$, of the S-plane. Figure 4.2#k shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+2S+2)}{(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.2#L, the multiple open-loop poles are located at -3, and the complex conjugate pair of open-loop zeros at $-1 \pm j1$, of the S-plane. Figure 4.2# l shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+8S+20)}{(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.2#M, the multiple open-loop poles are located at -3, and the complex conjugate pair of open-loop zeros at $-4 \pm j2$, of the S-plane. Figure 4.2#m shows the relevant graphs of G(S) which correspond to two cases of time delays: 0.2 and 1.0 seconds.

Combined (RL) and (SP) Graphs For Sets Of Second-Order Plant Configurations*

The graphic knowledge-base for second-order plants contains even a higher degree of information complexity than for the first-order. Hence, for easier comparative study, some of the plants' sensitivity profiles have been combined into groups of family sets of 3-d plot. Each family set is rotated around the sensitivity axis to give several viewing angles of perspective. Only two of these views have been captured and shown

* The combined plants represent examples some of which are not included in the concise knowledge-base given in appendix B.

here namely: family-set of sensitivity versus the real-axis and family-set of sensitivity versus the imaginary-axis of the S-domain.

The following combined graphs represent a few examples of second-order family sets selected mainly for illustration purposes:

Figure 4.2 shows the sensitivity profiles (sensitivity versus real & sensitivity versus imaginary) of a set of second-order plant configurations having the transfer functions: $G_{10}(S) = \frac{K}{(S+2)(S+1)}$, $G_{11}(S) = \frac{K}{(S+1.5)(S+1)}$ and $G_{12}(S) = \frac{K}{(S+5)(S+1)}$.

Figure 4.3 shows the sensitivity profiles (sensitivity versus real & sensitivity versus imaginary) of a set of second-order plant configurations having the transfer functions: $G_{13}(S) = \frac{K}{(S^2+2S+1.25)}$, $G_{14}(S) = \frac{K}{(S^2+2S+1.062)}$ and $G_{15}(S) = \frac{K}{(S^2+2S+5)}$.

Figure 4.4 shows the sensitivity profiles (sensitivity versus real & sensitivity versus imaginary) of a set of second-order plant configurations having the transfer functions: $G_{16}(S) = \frac{K(S+1.5)}{(S^2+3S+2)}$, $G_{17}(S) = \frac{KS}{(S^2+3S+2)}$, $G_{18}(S) = \frac{K(S+3)}{(S^2+3S+2)}$, $G_{19}(S) = \frac{K(S+5)}{(S^2+3S+2)}$ and $G_{20}(S) = \frac{K(S-3)}{(S^2+3S+2)}$.

Figure 4.5 shows the sensitivity profiles (sensitivity versus real & sensitivity versus imaginary) of a set of second-order plant configurations having the transfer functions: $G_{21}(S) = \frac{K(S+1)}{(S^2+2S+1.25)}$, $G_{22}(S) = \frac{K(S+1.5)}{(S^2+2S+1.25)}$, $G_{23}(S) = \frac{K(S+0.5)}{(S^2+2S+1.25)}$, $G_{24}(S) = \frac{K(S+5)}{(S^2+2S+1.25)}$ and $G_{25}(S) = \frac{K(S-3)}{(S^2+2S+1.25)}$.

Figure 4.6 shows the sensitivity profiles (sensitivity versus real & sensitivity versus imaginary) of a set of second-order plant configurations having the transfer functions:

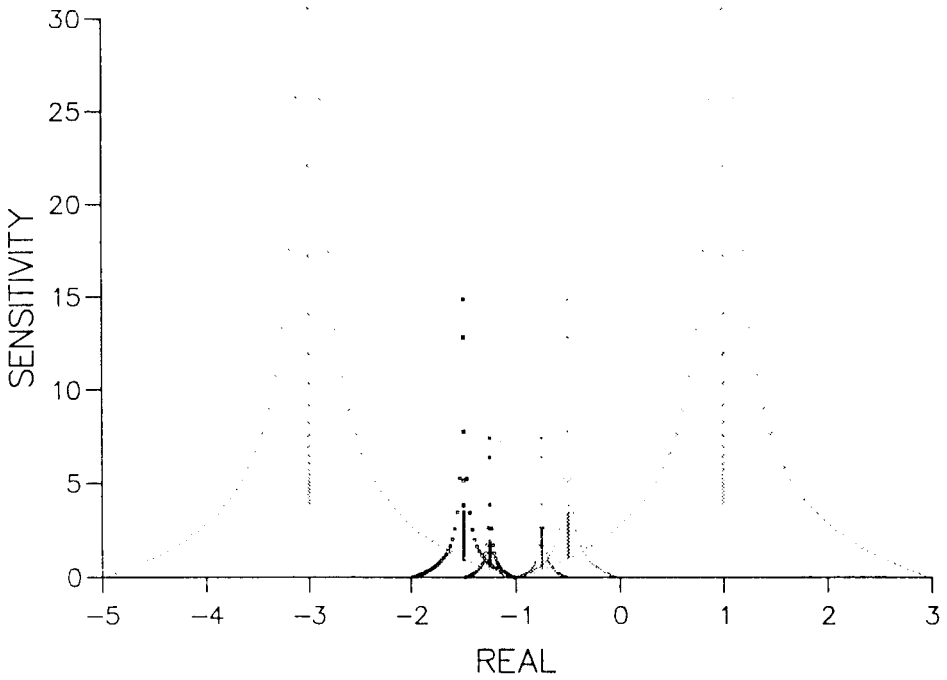
$G_{26}(S) = \frac{KS(S+1.5)}{(S^2+3S+2)}$, $G_{27}(S) = \frac{K(S+1.5)(S+3)}{(S^2+3S+2)}$, $G_{28}(S) = \frac{K(S+1.25)(S+1.75)}{(S^2+3S+2)}$, $G_{29}(S) = \frac{K(S+3)(S+5)}{(S^2+3S+2)}$ and $G_{30}(S) = \frac{KS(S-2)}{(S^2+3S+2)}$.

Figure 4.7 shows the sensitivity profiles (sensitivity versus real & sensitivity versus imaginary) of a set of second-order plant configurations having the transfer functions: $G_{31}(S) = \frac{KS(S+1)}{(S^2+2S+1.25)}$, $G_{32}(S) = \frac{KS(S+3)}{(S^2+2S+1.25)}$, $G_{33}(S) = \frac{K(S+1)(S+2)}{(S^2+2S+1.25)}$, $G_{34}(S) = \frac{K(S+1)(S+5)}{(S^2+2S+1.25)}$ and $G_{35}(S) = \frac{K(S^2+2S+1)}{(S^2+2S+1.25)}$.

Figure 4.8 shows the sensitivity profiles (sensitivity versus real & sensitivity versus imaginary) of a set of second-order plant configurations having the transfer functions: $G_{36}(S) = \frac{K(S^2+10S+25.062)}{(S^2+2S+1.25)}$, $G_{37}(S) = \frac{K(S^2+10S+27.25)}{(S^2+2S+1.25)}$ and $G_{38}(S) = \frac{K(S^2+10S+37.25)}{(S^2+2S+1.25)}$.

Figure 4.9 shows the sensitivity profiles (sensitivity versus real & sensitivity versus imaginary) of a set of second-order plant configurations having the transfer functions: $G_{39}(S) = \frac{K(S^2+2S+1.062)}{(S^2+2S+1.25)}$, $G_{40}(S) = \frac{K(S^2+2S+3.25)}{(S^2+2S+1.25)}$ and $G_{41}(S) = \frac{K(S^2+2S+13.25)}{(S^2+2S+1.25)}$.

Figure 4.10 shows the sensitivity profiles (sensitivity versus real & sensitivity versus imaginary) of a set of second-order plant configurations having the transfer functions: $G_{42}(S) = \frac{K(S^2-6S+9.062)}{(S^2+2S+1.25)}$, $G_{43}(S) = \frac{K(S^2-6S+11.25)}{(S^2+2S+1.25)}$ and $G_{44}(S) = \frac{K(S^2-6S+21.25)}{(S^2+2S+1.25)}$.



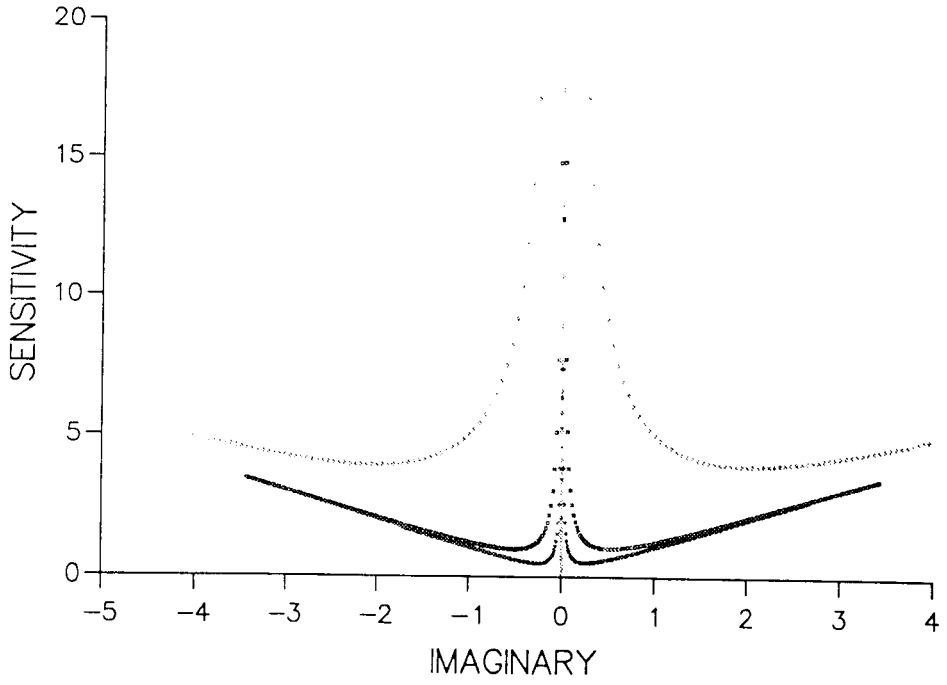
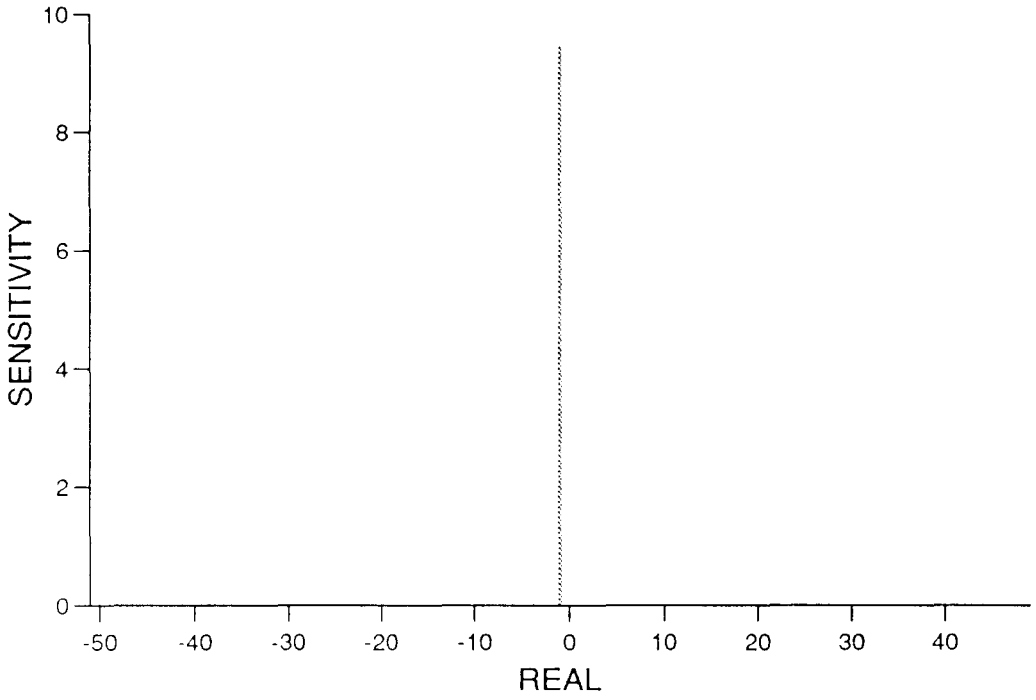


Figure 4.2 Two Views Of Combined Sensitivity Profiles For A Set Of Second-Order Plant's Configurations.



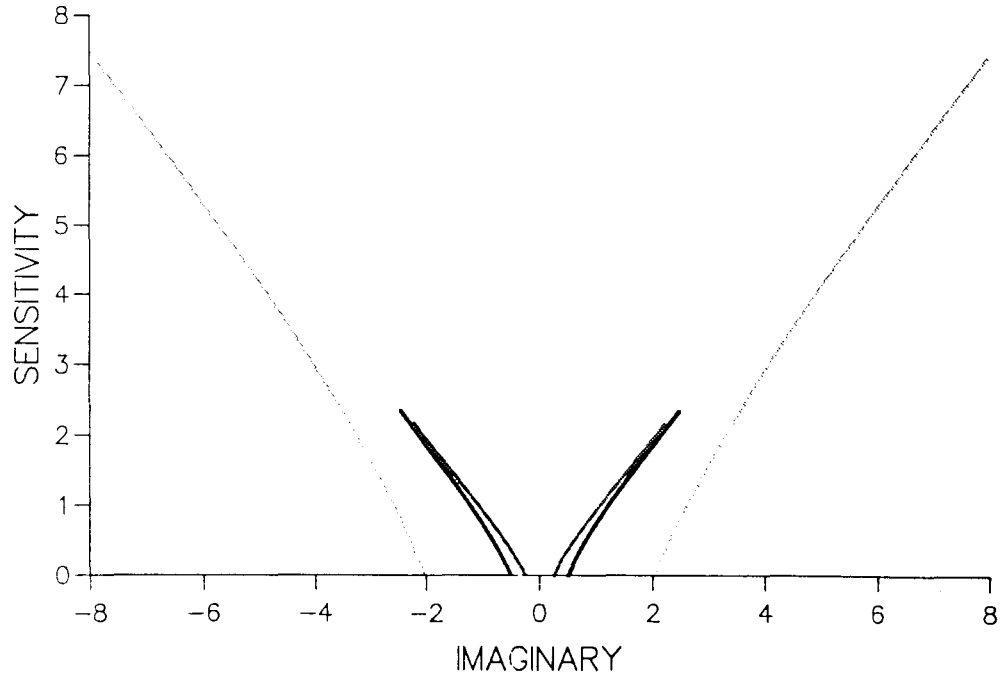


Figure 4.3 Two Views Of Combined Sensitivity Profiles For A Set Of Second-Order Plant's Configurations.

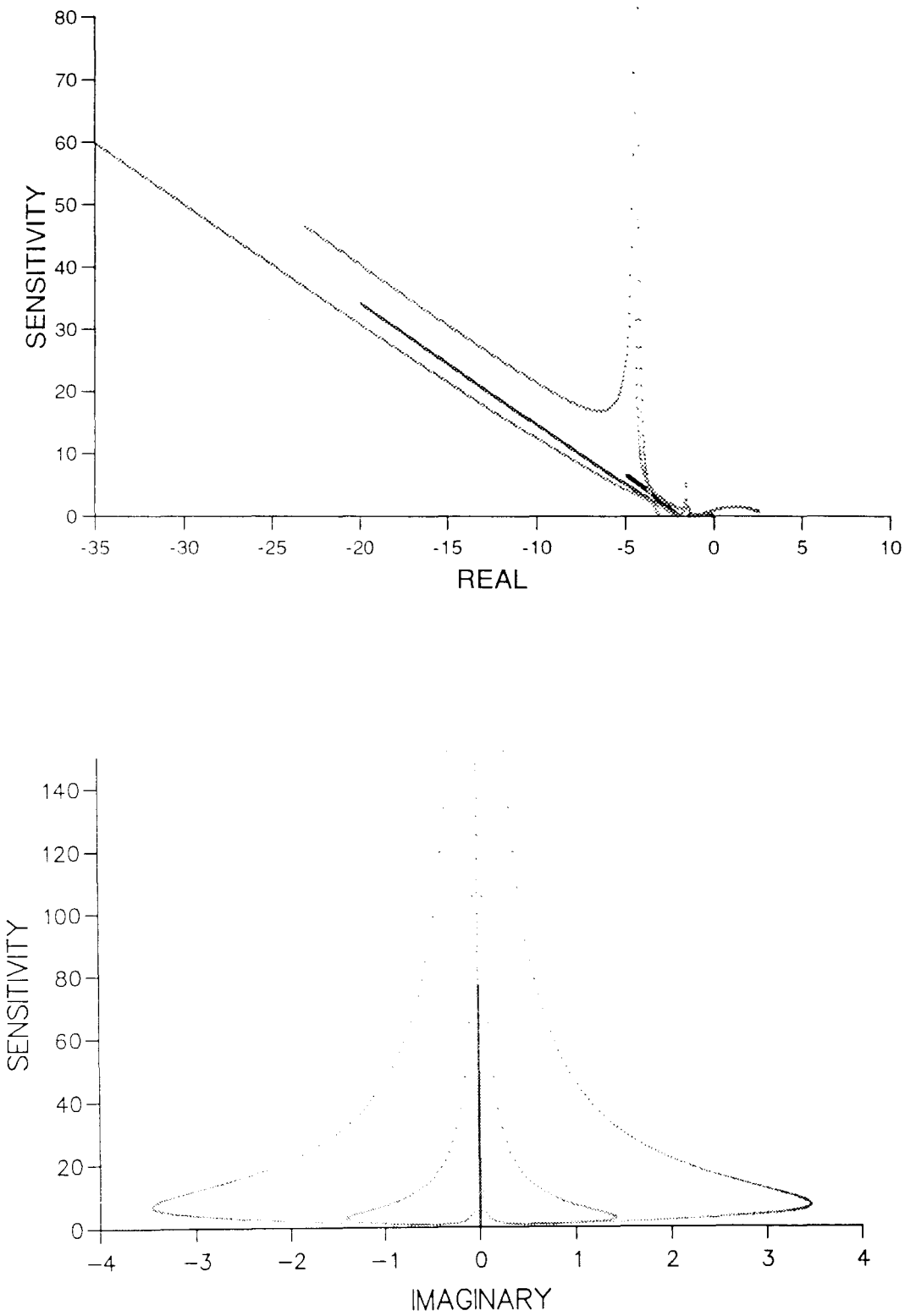


Figure 4.4 Two Views Of Combined Sensitivity Profiles For A Set Of Second-Order Plant's Configurations.

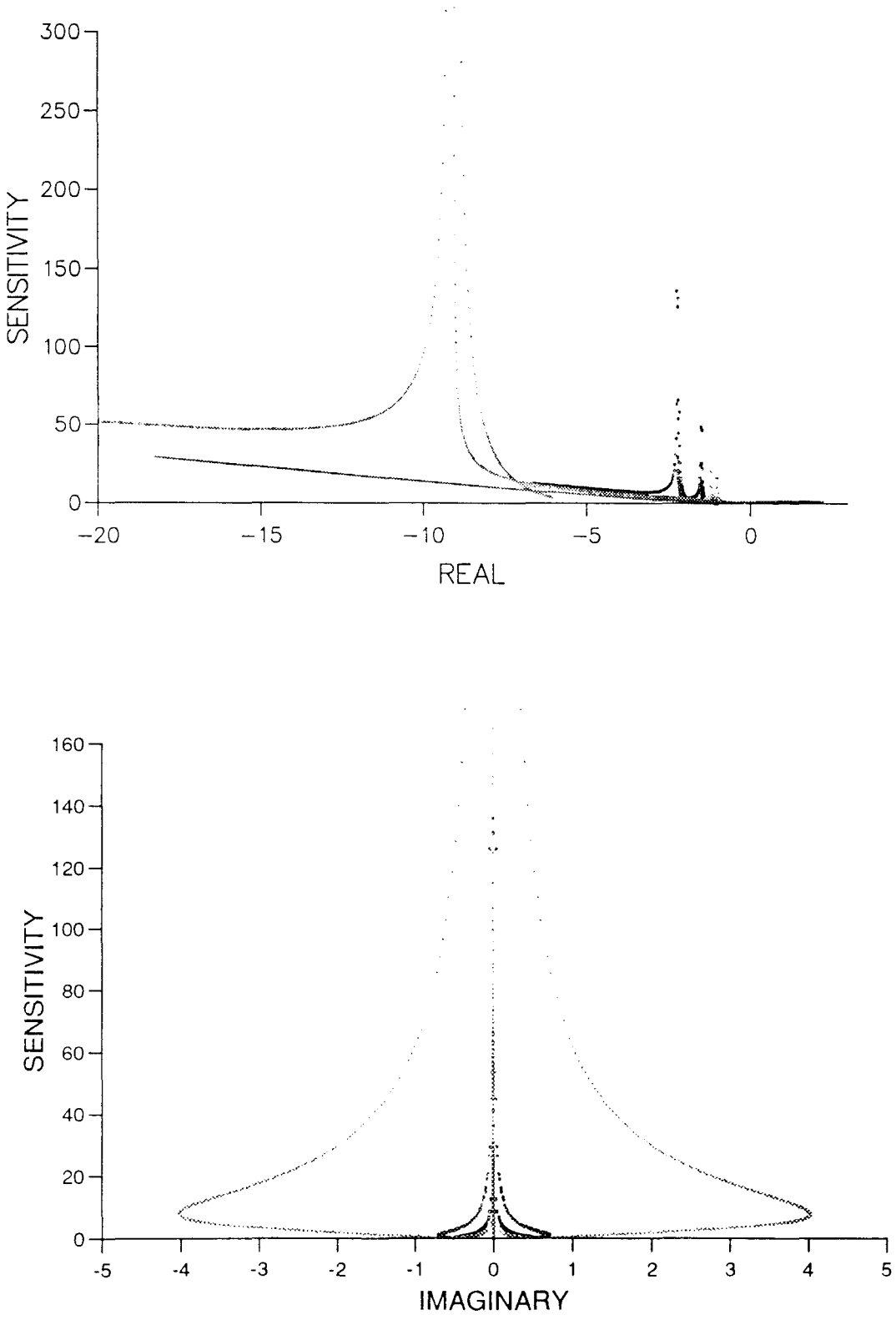


Figure 4.5 Two Views Of Combined Sensitivity Profiles For A Set Of Second-Order Plant's Configurations.

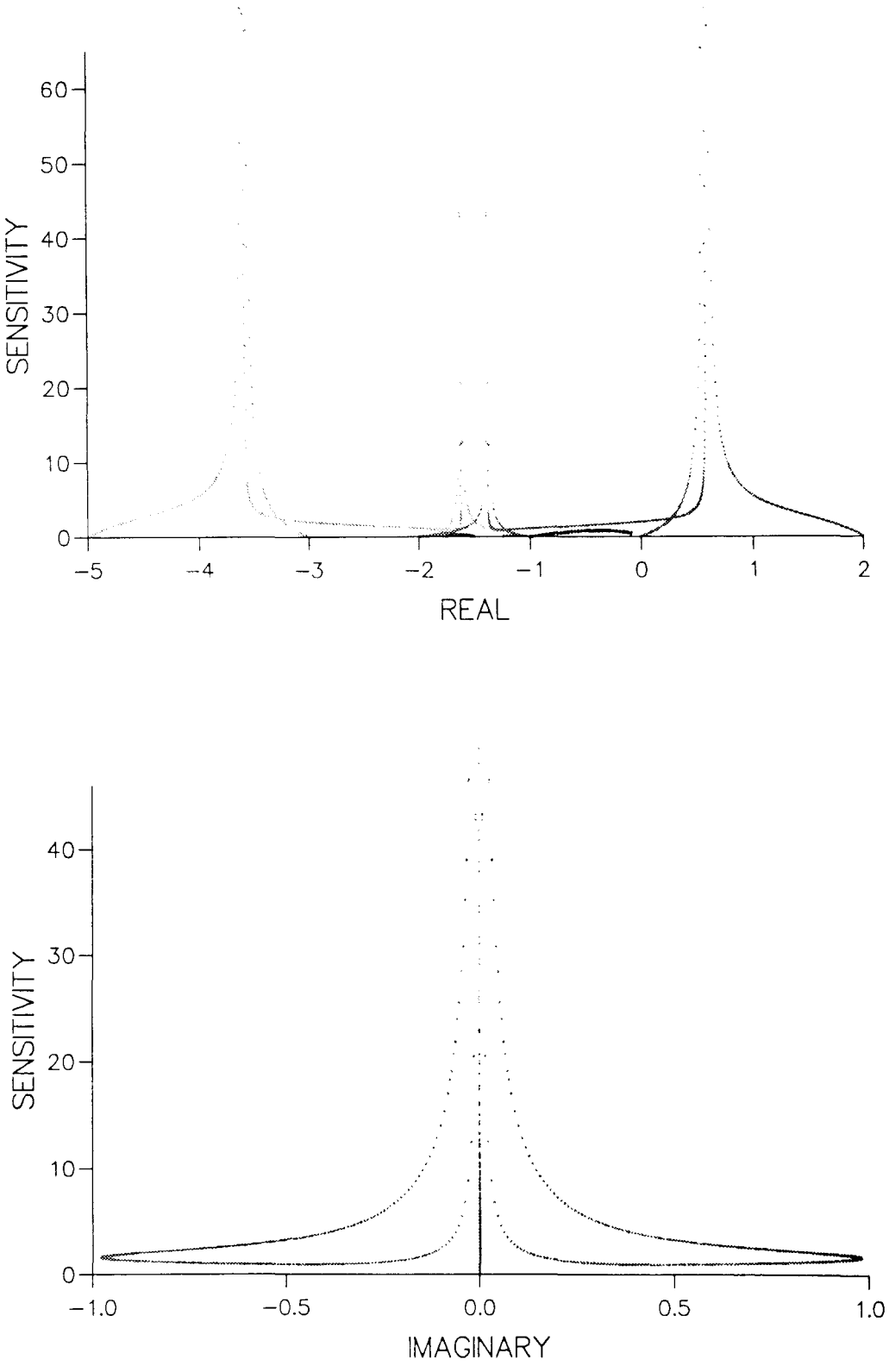


Figure 4.6 Two Views Of Combined Sensitivity Profiles For A Set Of Second-Order Plant's Configurations.

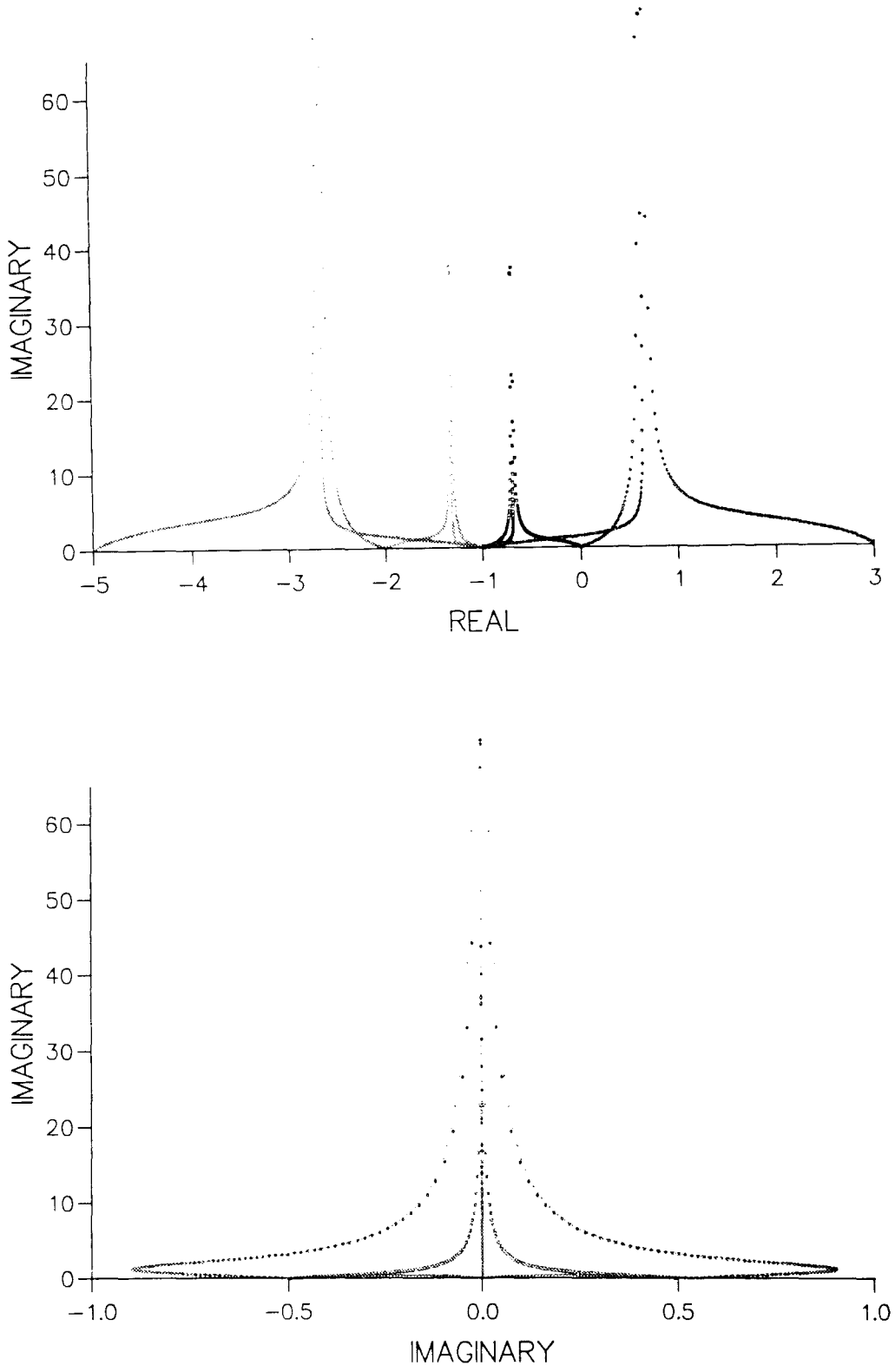
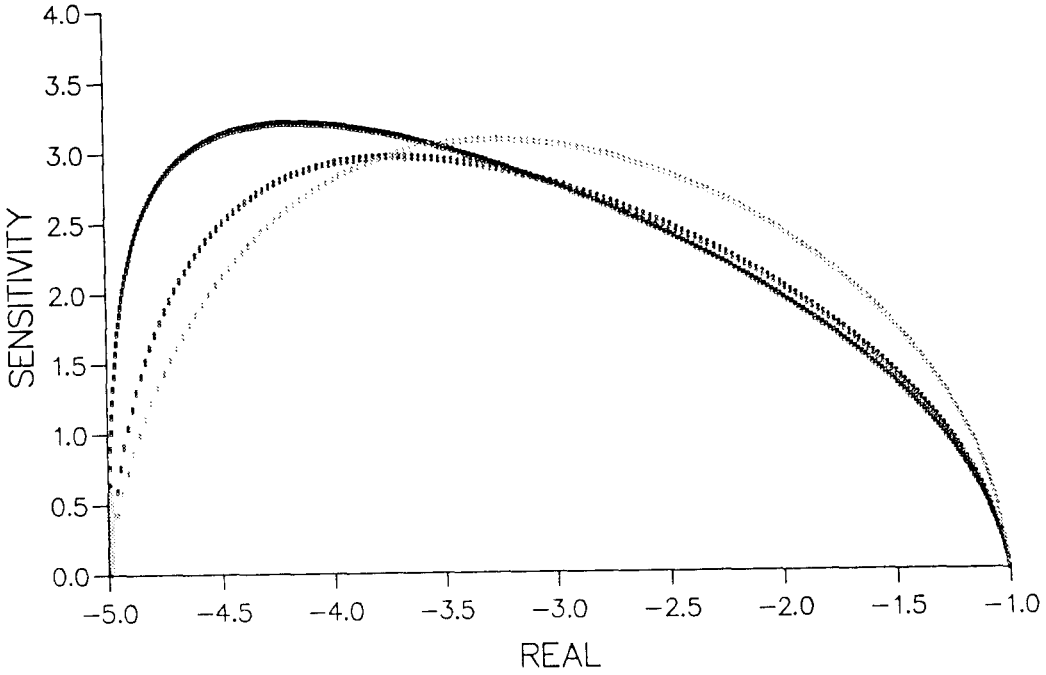


Figure 4.7 Two Views Of Combined Sensitivity Profiles For A Set Of Second-Order Plant's Configurations.



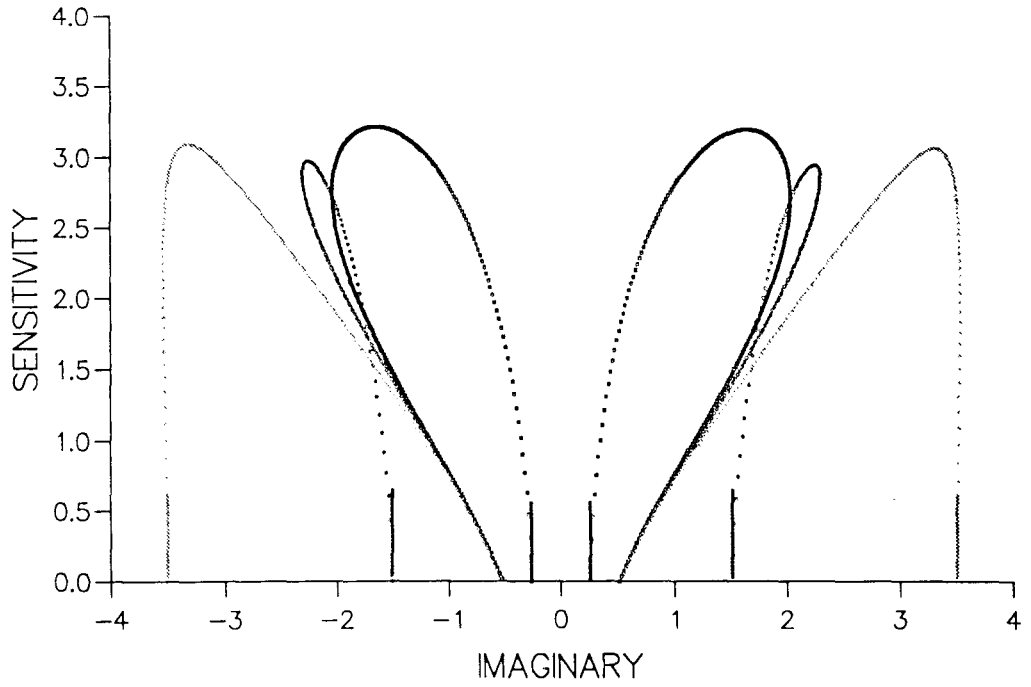


Figure 4.8 Two Views Of Combined Sensitivity Profiles For A Set Of Second-Order Plant's Configurations.

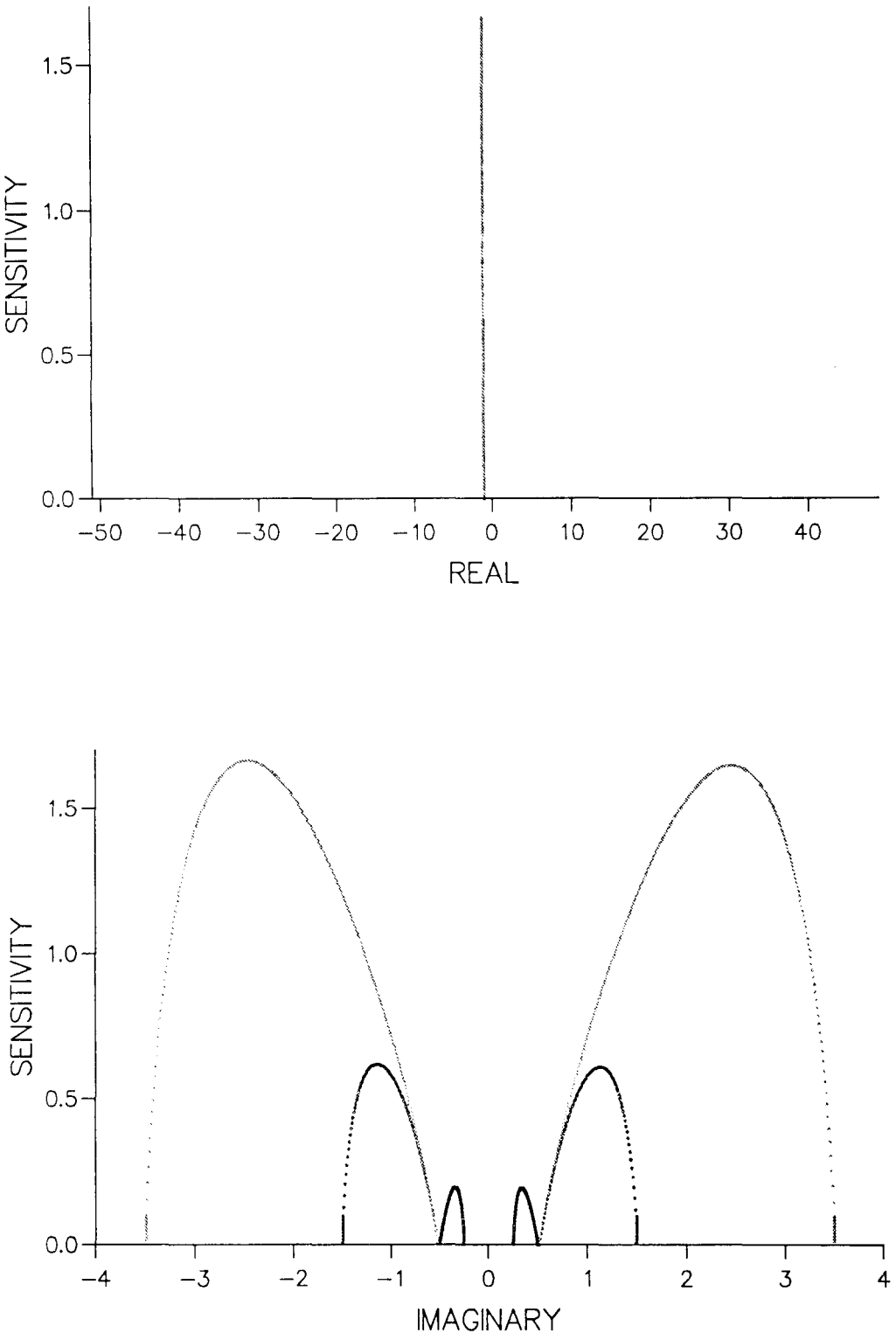


Figure 4.9 Two Views Of Combined Sensitivity Profiles For A Set Of Second-Order Plant's Configurations.

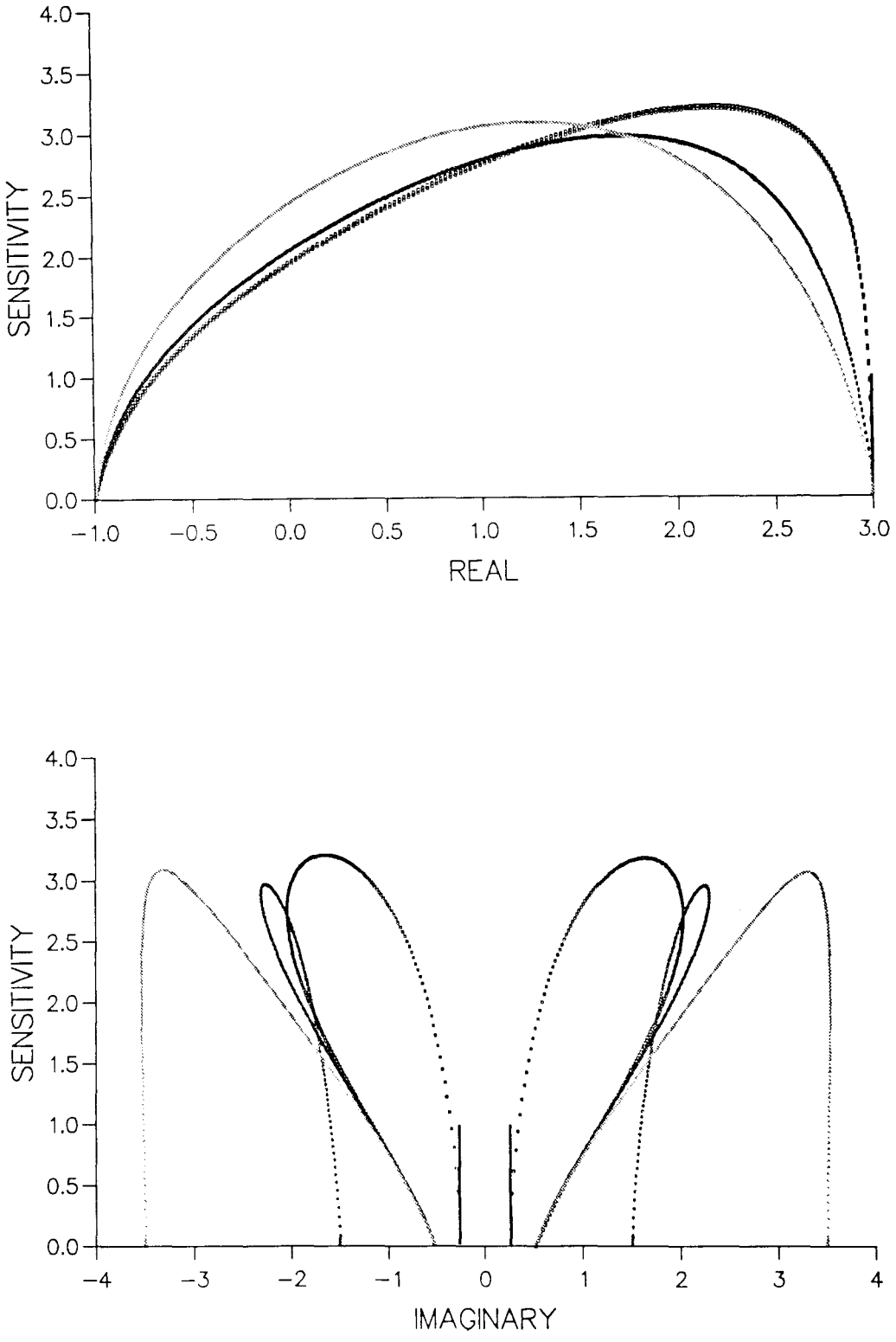


Figure 4.10 Two Views Of Combined Sensitivity Profiles For A Set Of Second-Order Plant's Configurations.

For plants of orders higher than the second, the number of possibilities of their configurations are too numerous to find a representative number. Therefore, only a sample of higher-order plant configurations with zero delay have been randomly selected for illustration purposes as outlined in the subsequent subsections.

4.3.3 Third-Order Plants

*A deterministic plant with a transfer function, $G(S) = \frac{K}{(S+3)(S^2+2S+1)}$, (RL) and (SP) are illustrated in figure 4.3A, the open-loop poles are located at: -3 and a second-order pole at -1, and three open-loop zeros at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K}{(S+1)(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.3B, the open-loop poles are located: at -1 and a second-order pole at -3, and three open-loop zeros at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K}{S(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.3C, the open-loop poles are located: at the origin and a second-order pole at -3, and three open-loop zeros: at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S+2)}{S(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.3D, the open-loop poles are located: at the origin and a second-order pole at -3, and the open-loop zeros: two at infinity and one at -2, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S+1)}{S(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.3E, the open-loop poles are located: at the origin and a

second-order pole at -3, and the open-loop zeros: two at infinity and one at -1, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S+2.5)}{S(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.3F, the open-loop poles are located: at the origin and a second-order pole at -3, and the open-loop zeros: two at infinity and one at -2.5, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S+2.7)}{S(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.3G, the open-loop poles are located: at the origin and a second-order pole at -3, and the open-loop zeros: two at infinity and one at -2.7, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{KS}{(S+2)(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.3H, the open-loop poles are located: at -2 and a second-order pole at -3, and the open-loop zeros: two at infinity and one at the origin, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{KS}{(S+2.8)(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.3I, the open-loop poles are located: at -2.8 and a second-order pole at -3, and the open-loop zeros: two at infinity and one at the origin, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S+3)}{S^2(S+2)}$, (RL) and (SP) are illustrated in figure 4.3J, the open-loop poles are located: one at -2 and a second-order pole at the origin, and the open-loop zeros: two at infinity and one at -3, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S+4)}{(S+3)(S^2+2S+1)}$, (RL) and (SP) are illustrated in figure 4.3K, the open-loop poles are located: one at -3 and a second-order pole at -1, and the open-loop zeros: two at infinity and one at -4, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S+2)}{S^2(S+3)}$, (RL) and (SP) are illustrated in figure 4.3L, the open-loop poles are located: one at -3 and a second-order pole at the origin, and the open-loop zeros: two at infinity and one at -2, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K}{S^3}$, (RL) and (SP) are illustrated in figure 4.3M, a third-order open-loop pole is located at the origin, and the open-loop zeros at infinity , of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K}{(S+2)(S^2+4S+4)}$, (RL) and (SP) are illustrated in figure 4.3N, a third-order open-loop pole is located at -2, and the open-loop zeros at infinity , of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S+3)}{(S+2)(S^2+4S+4)}$, (RL) and (SP) are illustrated in figure 4.3O, a third-order open-loop pole is located at -2, and the open-loop zeros: two at infinity and one at -3 , of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S+1)}{(S+3)(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.3P, a third-order open-loop pole is located at -3, and the open-loop zeros: two at infinity and one at -1, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+2S+1)}{(S+3)(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.3Q, a third-order open-loop pole is located at -3, and the open-loop zeros: multiple at -1 and one at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+2S+5)}{(S+3)(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.3R, a third-order open-loop pole is located at -3, and the open-loop zeros: a complex conjugate pair at $-1 \pm j2$ and one at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{KS(S^2+2S+5)}{(S+3)(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.3S, a third-order open-loop pole is located at -3, and the open-loop zeros: a complex conjugate pair at $-1 \pm j2$ and one at the origin, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{KS(S^2-2S+5)}{(S+3)(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.3T, a third-order open-loop pole is located at -3, and the open-loop zeros: a complex conjugate pair at $+1 \pm j2$ and one at the origin, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{KS^3}{(S+3)(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.3U, a third-order open-loop pole is located at -3, and the multiple open-loop zeros at the origin, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S-3)(S^2-6S+9)}{(S+3)(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.3V, a third-order open-loop pole is located at -3, and the multiple open-loop zeros at +3, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S+1)(S+2)}{S(S^2+1)}$, (RL) and (SP) are illustrated in figure 4.3W, the open-loop poles are located: a single at the origin and a complex conjugate pair at $0 \pm j1$, and the open-loop zeros at -1, -2 and at infinity, of the S-plane.

4.3.4 Fourth-Order Plants

*A deterministic plant with a transfer function, $G(S) = \frac{K}{(S+3)(S+!)(S^2+2S+1)}$, (RL) and (SP) are illustrated in figure 4.4A, the open-loop poles are located: a single at -3, and a third-order at -1, and four open-loop zeros at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K}{(S+1)(S+3)(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.4B, the open-loop poles are located: a single at -1, and a third-order at -3, and four open-loop zeros at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S+2)}{S(S+3)(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.4C, the open-loop poles are located: a single at the origin, and a third-order at -3, and the open-loop zeros: three at infinity and one at -2, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{KS}{(S+1)(S+3)(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.4D, the open-loop poles are located: a single at -1, and a third-order at -3, and the open-loop zeros: three at infinity and one at the origin, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S+2)}{S^3(S+3)}$, (RL) and (SP) are illustrated in figure 4.4E, the open-loop poles are located: a single at -3, and a third-order at the origin, and the open-loop zeros: three at infinity and one at -2, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S+3)}{S^3(S+2)}$, (RL) and (SP) are illustrated in figure 4.4F, the open-loop poles are located: a single at -2, and a third-order at the origin, and the open-loop zeros: three at infinity and one at -3, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K}{(S^2+2S+1)(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.4G, the open-loop poles are located: two second-order poles at -1 and at -3, and four open-loop zeros at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+8S+16)}{(S^2+4S+4)(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.4H, the open-loop poles are located: two second-order poles at -2 and at -3, and the open-loop zeros: a second-order at -4 and two at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K}{(S^2+2S+1)(S^2-2S+1)}$, (RL) and (SP) are illustrated in figure 4.4I, the open-loop poles are located: two second-order poles at -1 and at +1, and four open-loop zeros at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K}{(S^2+4S+8)(S^2+4S+8)}$, (RL) and (SP) are illustrated in figure 4.4J, the second-order complex conjugate pairs

of open-loop poles are located at -2 $\pm$ j2, and four open-loop zeros at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K}{(S^2+4S+4)(S^2+2S+5)}$, (RL) and (SP) are illustrated in figure 4.4k, the open-loop poles are located: a second-order at -2 and a complex conjugate pair at -1 $\pm$ j2, and four open-loop zeros at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K}{(S^2+4S+4)(S^2+4)}$, (RL) and (SP) are illustrated in figure 4.4L, the open-loop poles are located: a second-order pole at -2 and a complex conjugate pair at 0 $\pm$ j2, and four open-loop zeros at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{KS^2}{(S^2+4S+8)(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.4M, the open-loop poles are located: a second-order pole at -3 and a complex conjugate pair at -2 $\pm$ j2 and the open-loop zeros: a second-order at the origin and two at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2-1)}{(S^2+4S+8)(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.4N, the open-loop poles are located: a second-order pole at -3 and at a complex conjugate pair at -2 $\pm$ j2, and the open-loop zeros: a second-order at +1 and two at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K}{(S^2+4S+8)(S^2+2S+1)}$, (RL) and (SP) are illustrated in figure 4.4 O, the open-loop poles are located: a second-order pole at -1 and at a complex conjugate pair at -2 $\pm$ j2, and the open-loop zeros: four at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K}{S^2(S^2+4S+8)}$, (RL) and (SP) are illustrated in figure 4.4P , the open-loop poles are located: a second-order pole at the origin and a complex conjugate pair at $-2 \pm j2$, and the open-loop zeros: four at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S+3)}{S^2(S^2+4S+8)}$, (RL) and (SP) are illustrated in figure 4.4Q , the open-loop poles are located: a second-order pole at the origin and a complex conjugate pair at $-2 \pm j2$, and the open-loop zeros: one at -3 and three at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+4S+8)}{S^2(S^2+6S+13)}$, (RL) and (SP) are illustrated in figure 4.4R , the open-loop poles are located: a second-order pole at the origin and a complex conjugate pair at $-3 \pm j2$, and the open-loop zeros: at $-2 \pm j2$ and two at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+6S+13)}{S^2(S^2+4S+8)}$, (RL) and (SP) are illustrated in figure 4.4S , the open-loop poles are located: a second-order pole at the origin and a complex conjugate pair at $-2 \pm j2$, and the open-loop zeros: at $-3 \pm j2$ and two at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+4S+8)}{(S^2+4)(S^2+6S+9)}$, (RL) and (SP) are illustrated in figure 4.4T , the open-loop poles are located: a second-order pole at -3 and a complex conjugate pair at $0 \pm j2$, and the open-loop zeros: at $-2 \pm j2$ and two at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+2S+5)}{(S^2-2S+5)(S^2+6S+9)}$,

(RL) and (SP) are illustrated in figure 4.4U , the open-loop poles are located: a second-order pole at -3 and a complex conjugate pair at $+1 \pm j2$, and the open-loop zeros: at $-1 \pm j2$ and two at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+4)}{(S^2+4S+8)(S^2+6S+9)}$,

(RL) and (SP) are illustrated in figure 4.4V , the open-loop poles are located: a second-order pole at -3 and a complex conjugate pair at $-2 \pm j2$, and the open-loop zeros: at $0 \pm j2$ and two at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2-2S+5)}{(S^2+4S+8)(S^2+6S+9)}$,

(RL) and (SP) are illustrated in figure 4.4W , the open-loop poles are located: a second-order pole at -3 and a complex conjugate pair at $-2 \pm j2$, and the open-loop zeros: at $+1 \pm j2$ and two at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+4S+4)}{(S^2+2S+5)(S^2+6S+9)}$,

(RL) and (SP) are illustrated in figure 4.4X , the open-loop poles are located: a second-order pole at -3 and a complex conjugate pair at $-1 \pm j2$, and the open-loop zeros: a second-order at -2 and two at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+2S+1)}{(S^2+4S+8)(S^2+6S+9)}$,

(RL) and (SP) are illustrated in figure 4.4Y , the open-loop poles are located: a second-order pole at -3 and a complex conjugate pair at $-2 \pm j2$, and the open-loop zeros: a second-order at -1 and two at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S+4)}{(S^2+4S+8)(S^2+4S+8)}$,

(RL) and (SP) are illustrated in figure 4.4Z , the open-loop poles are located: a

second-order complex conjugate pair at -2± j2, and the open-loop zeros: a single at -4 and three at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+6S+10)}{(S^2+4S+8)(S^2+4S+8)}$, (RL) and (SP) are illustrated in figure 4.4#A , the open-loop poles are located: a second-order complex conjugate pair at -2± j2, and the open-loop zeros: at -3± j1 and two at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+6S+13)(S^2+6S+13)}{(S^2+2S+5)(S^2+2S+5)}$, (RL) and (SP) are illustrated in figure 4.4#B , the open-loop poles are located: a second-order complex conjugate pair at -1± j2, and the open-loop zeros: a second-order complex conjugate pair at -3± j2, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+6S+9.0625)(S^2+6S+9.0625)}{(S^2+2S+5)(S^2+2S+5)}$, (RL) and (SP) are illustrated in figure 4.4#C , the open-loop poles are located: a second-order complex conjugate pair at -1± j2, and the open-loop zeros: a second-order complex conjugate pair at -3± j0.5, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+4)(S^2+4)}{(S^2+6S+13)(S^2+6S+13)}$, (RL) and (SP) are illustrated in figure 4.4#D , the open-loop poles are located: a second-order complex conjugate pair at 0± j2, and the open-loop zeros: a second-order complex conjugate pair at -3± j2, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2-4S+8)(S^2-4S+8)}{(S^2+6S+13)(S^2+6S+13)}$, (RL) and (SP) are illustrated in figure 4.4#E , the open-loop poles are located: a

second-order complex conjugate pair at -3± j2, and the open-loop zeros: a second-order complex conjugate pair at +2± j2, of the S-plane.

4.3.5 Fifth-Order Plant

*A deterministic plant with a transfer function, $G(S) = \frac{K}{(S+3)(S^2+2S+5)(S^2+2S+5)}$, (RL) and (SP) are illustrated in figure 4.5A , the open-loop poles are located: a single at -3 and a second-order complex conjugate pair at -1± j2, and five open-loop zeros at infinity, of the S-plane.

4.3.6 Sixth-Order Plants

*A deterministic plant with a transfer function, $G(S) = \frac{K}{(S^3+3S^2+3S+1)(S^3-3S^2+3S-1)}$, (RL) and (SP) are illustrated in figure 4.6A , the open-loop poles are located: a third-order at -1 and at +1, and six open-loop zeros at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K}{(S^3+3S^2+3S+1)(S^3-3S^2+3S-1)}$, (RL) and (SP) are illustrated in figure 4.6B , the open-loop poles are located: a third-order at -2 and at -3, and open-loop zeros: at -1 ±j2 and at -3.5 ±j2 and two at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K}{(S^2+6S+10)(S^2+4S+8)(S^2+4S+8)}$, (RL) and (SP) are illustrated in figure 4.6C , the

open-loop poles are located: a second-order complex conjugate pair at -2±j2 and a complex conjugate pair at -3±j1 , and six open-loop zeros at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+6S+18)(S^2+6S+18)}{(S^2+2S+5)(S^2+2S+5)(S^2+6S+10)}$, (RL) and (SP) are illustrated in figure 4.6D , the open-loop poles are located: a second-order complex conjugate pair at -1±j2 and a complex conjugate pair at -3±j1 , and open-loop zeros: a second-order complex conjugate pair at -3±j3 and two at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K}{(S^2+4S+8)(S^2+4S+8)(S^2+8S+16)}$, (RL) and (SP) are illustrated in figure 4.6E , the open-loop poles are located: a second-order complex conjugate pair at -2±j2 and a second-order at -4 , and six open-loop zeros at infinity, of the S-plane.

*A deterministic plant with a transfer function, $G(S) = \frac{K(S^2+2S+2)}{(S^2+4S+8)(S^2+4S+8)(S^2+8S+16)}$, (RL) and (SP) are illustrated in figure 4.6F , the open-loop poles are located: a second-order complex conjugate pair at -2±j2 and a second-order at -4 , and open-loop zeros: at a complex conjugate pair-1± j1 and four at infinity, of the S-plane.

4.3.7 Eighth-Order Plant

*A deterministic plant with a transfer function, $G(S) = \frac{K}{(S^2+4S+8)(S^2+4S+8)(S^2+6S+10)(S^2+6S+10)}$, (RL) and (SP) are illustrated in figure 4.8A , the open-loop poles are located: a second-order complex conjugate pair at

-2±j2 and a complex conjugate pair at -3±j1 , and Eight open-loop zeros at infinity, of the S-plane.

4.4 Conclusions

The objective of the work in this chapter has been primarily targeted to the validation of the S-B algorithm implementation and to the evaluation of all aspects of its designed performance. To this end, an extensive programme of experiments were conducted using a fairly comprehensive range of feed-back models of deterministic plant configurations and delays. Easily derivable theoretical results for first-order and second-order models were used as standard bench-marks for validation of the algorithm performance and for assessing its accuracy. These were found to be good. The second stage of the programme of evaluation was in the form of comparing the algorithm performance with that obtained from a commercial CAD system using examples of higher-order deterministic plant models. The results of these tests have revealed that the developed S-B CAD system is far superior over its commercial counterpart in terms of accuracy, speed, range of automated facilities, accessibility to a vast array of the modelled system behaviour either in detailed step-by-step viewing mode or in continuously updating mode.

These advantages were exploited in generating a graphic knowledge-base classification of feed-back systems incorporating deterministic plants of diverse configurations and delays, as described in section 4.3. The knowledge-base reveals a relatively comprehensive insight into system's behaviour which may assert the value of the S-B algorithm as an efficient CAD tool, the extent of which will be examined in connection with the work in chapters 5, 6 & 7.

Chapter 5

Deployment Of The S-B Algorithm As A CAD Tool For Controlling Deterministic Plants

5.1 Introduction

This chapter conducts a case study of a real-life plant with the objective to demonstrate certain aspects of the CAD facility of the algorithm for optimum controller design purposes and to highlight its significance as both an efficient and an effective tool for the designer/analyst. The case study refers to the design of a typical autopilot control system when the aircraft dynamics including its actuator are approximated by a deterministic model.

In chapter 4 the KBSC of deterministic plants or systems of increasing orders has shown that the sensitivity-based algorithm provides interactive and highly automated environment where 2-d and 3-d functional diagrams, which combine performance and robustness information, are displayed rapidly as a moving picture. In principle, the incorporation of a controller pole/zero configuration selected from the KBSC with the plant in a closed-loop system, then viewing graphically the effects on both the performance and robustness can be carried out almost continuously in-time. This allows the designer to place the controller pole /zero pattern, which optimises the overall system characteristics, rapidly.

Apart from the efficient computation and almost instantaneous displays of the system root-loci accompanied with the root- sensitivities, the algorithm allows the superposition of performance specification loci (in terms of overshoot, settling time, band width, time to the peak), as well as sensitivity requirements (in the form of sensitivity threshold planes parallel to the complex-frequency plane) to guide the designer to the optimum controller configuration and to the optimum system's gain. The continuous 2-d and 3-d composite graphical displays, the fast response and user-friendly environment provided by the algorithm can determine the operating zone of the system centred on its optimum condition and yet meet the design specifications. This facility has a practical significance, since it caters for parameter variations in the system as will be dealt with in chapters 6 and 7, though is confined here to **only gain and delay variations.**

5.2 Case Study: Design Of A Robust Aircraft Auto Pilot*

A typical aircraft autopilot control system consists of electrical, mechanical, and hydraulic devices that move the flaps, elevators, fuel-flow controllers, and other components that cause the aircraft to vary its flight. Sensors provide information on velocity, heading rate of rotation, and other flight data. This information is combined with the desired flight characteristics (commands) electronically available to the autopilot. The autopilot should be able to fly the aircraft on a heading and under conditions set by pilot. The command often consists of a predetermined heading. Design often focuses on a forward-moving aircraft that moves somewhat up or down without moving right or left and without rolling (rotating the wingtips). Such a study is called **pitch-plane design** Figure 5.1 portrays an aircraft with the pitch angular rotation ' ϕ ' clearly indicated.

* Source: Modern Control Systems, Richard C. Dorf, Sixth Edition, 1992, PP. 599 section 11.8.

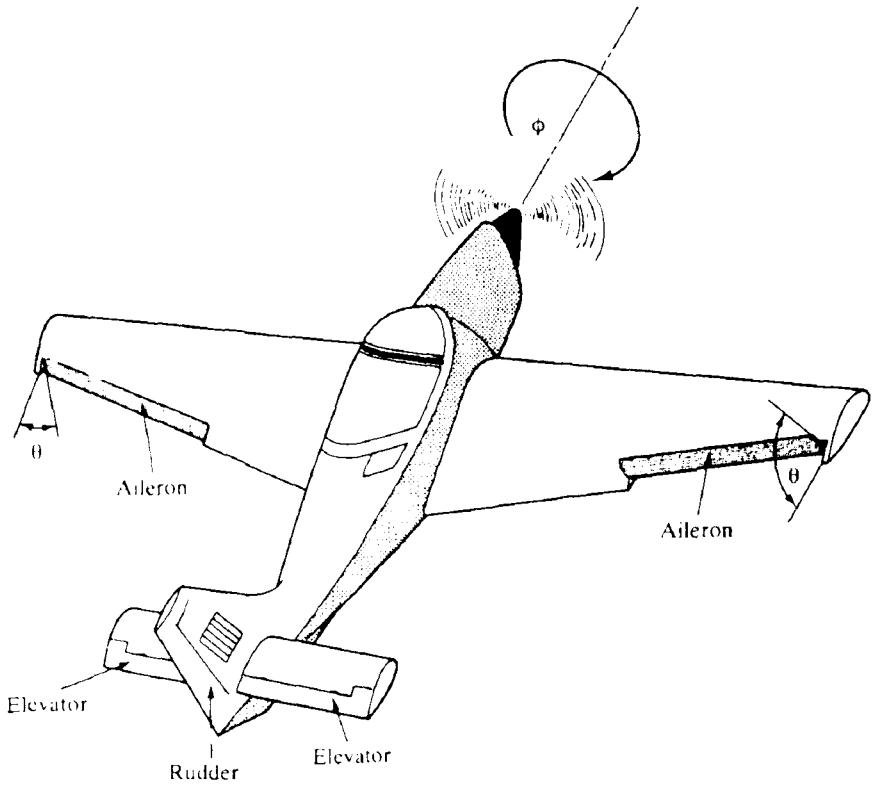


Figure 5.1 Portrait Of An Aircraft With The Pitch Angular Rotation " ϕ " Indicated

5.2.1 The Plant

In this case study, the aircraft pitch-plane dynamics and the actuator characteristics are represented approximately by a deterministic plant:

$$G(S) = \frac{K_p}{(S + \frac{1}{\tau_1})(S^2 + 2\xi_1\omega_1 S + \omega_1^2)} \qquad (5.1)$$

Where K_p is the plant gain = 1.0,

τ_1 is the time constant of the actuator =0.2 Sec.,

ω_1 is the undamped natural frequency =2.0 rad./s,

and ξ_1 is the damping ratio = 0.5

The pole/zero configuration of the plant in the S-plane is as follows:

* A real pole at S = -4.0
* A pair of complex conjugate poles at S = -1.0 $\pm$ j 1.73

The plant is assumed in the first place to have a zero delay. However, in order to widen the design problem to cover aspects encountered in other plants so as to investigate how the developed CAD tool can cope, it was decided to consider four cases of delays:

i) Plant with zero delay; $G_1(S) = G(S)\ e^{(-T_{d1} S)};\ T_{d1} = 0.00$ sec., (5.2)

ii) Plant with 0.05 sec. delay; $G_2(S) = G(S)\ e^{(-T_{d2} S)};\ T_{d2} = 0.05$ sec., (5.3)

iii) Plant with 0.1 sec. delay; $G_3(S) = G(S)\ e^{(-T_{d3} S)};\ T_{d3} = 0.10$ sec., (5.4)

iv) Plant with 0.2 sec. delay; $G_4(S) = G(S)\ e^{(-T_{d4} S)};\ T_{d4} = 0.20$ sec., (5.5)

5.2.2 Desired Specifications Of The Control System

Table 5.1 summarises the time-domain performance specifications for a unit-step target signal representing the pitch angle. It is also desired that:

* the steady-state error for a unit-step pitch rate should not exceed 0.6 rad. s^{-1},
* The band-width should be in the range of 2.5 to 3.5 rad. s^{-1}, and
* The sensitivity of the dominant closed-loop poles should be less than 2.0.

Table 5.1: Required Performance Specifications For A Unit-Step Target Signal

Steady-State Error of the Pitch Angle (rad.)	Percent overshoot	Settling time (sec.)	Rise time (sec.)
0.0	< 7.0	< 2.5	< 1.0

5.2.3 Performance Specifications Loci As An Aid To The Design Process

The design process principally involves the selection of the controller configuration, and the adjustment of the parameters (usually gain) of the control system in order to provide the desired response. Because control systems are inherently dynamic systems, the performance is invariably specified in terms of both the time response for a specified input target signal and the resulting steady-state error.

Real plants, however, have parameters which may vary with environmental conditions, difficult to measure accurately, or difficult to calculate with certainty. Furthermore, real plants are likely to be subjected to noise interference and disturbance signals. Therefore, specifications should include measures of roots and system's sensitivities to reflect the desired degrees of the control system robustness and its immunity to both noise and disturbances.

The specifications are seldom a rigid set of requirements, but rather a first attempt at listing a desired performance and quality assurance measure. Often in the course of any design, the specifications are revised in order to effect a compromise. In other words, the design process is inherently an iterative multi-decision process aiming to narrowing the choices of the controller configuration, on one hand, and to narrowing the operating zone of the control system centred on its optimum condition through selection of narrow ranges of parameters, on the other, to yield a robust design. Depending on the tolerable variance of the system's characteristics, the controller design may lead to either a fixed configuration or an adaptive type.

The transient performance of systems in response to a unit-step input target signal is well known to correlate to the systems' closed-loop poles locations in the S-plane as

well as to their associated residues. Such correlations are relatively easy and well defined for systems up to second-order particularly if the closed-loop zeros are located at infinity. For instance, a second-order closed-loop system having a pair of complex conjugate poles may be described by the normalised standard transfer function:

$$H(S) = \frac{\omega_n^2}{(S^2+2\xi\omega_n S+ \omega_n^2)} \; ; \; \xi < 1 \qquad (5.6)$$

with the poles located at $S = - \xi\omega_n \pm j\omega_n\sqrt{1-\xi^2}$ (5.7)

The transient (step-response) performance specifications can be derived in terms of the following four indices:

* The percent overshoot = 100 x $\exp-\left(\frac{\xi\pi}{\sqrt{1-\xi^2}}\right)$ (5.8)

* Time to the peak (as an approximate indication of the rise time)$=\left(\frac{\pi}{\omega_n\sqrt{1-\xi^2}}\right)$ sec. (5.9)

* Settling time = $\frac{\ln\left(\frac{1}{e_{tol}}\right)}{\xi\omega_n}$ sec. (5.10)

where e_{tol} is the system's tolerable error

* The band-width is approximately = ω_n rads. s^{-1}. (5.11)

These relationships are functions of the damping ratio ξ, the undamped frequency ω_n or a combination of both. Since the closed-loop poles (equation 5.7) are also functions of the same, it follows therefore that the four performance indices (equations 5.8 to 5.11) can be translated into four sets of loci in the S-plane for all possible locations of a pair of complex conjugate poles for pre-determined fixed values of any of these indices as shown in figure 5.2.

These performance specification loci can be per-calculated off-line and included in the CAD system as a knowledge-base for guiding design decisions relevant to closed-loop poles placing which optimise these performance indices.

These sets of loci, although relevant to a specific class of second-order systems as mentioned earlier, they nevertheless represent a useful design tool (in conjunction with others) for control systems of higher-order which can be approximated by a pair of dominant closed-loop poles.

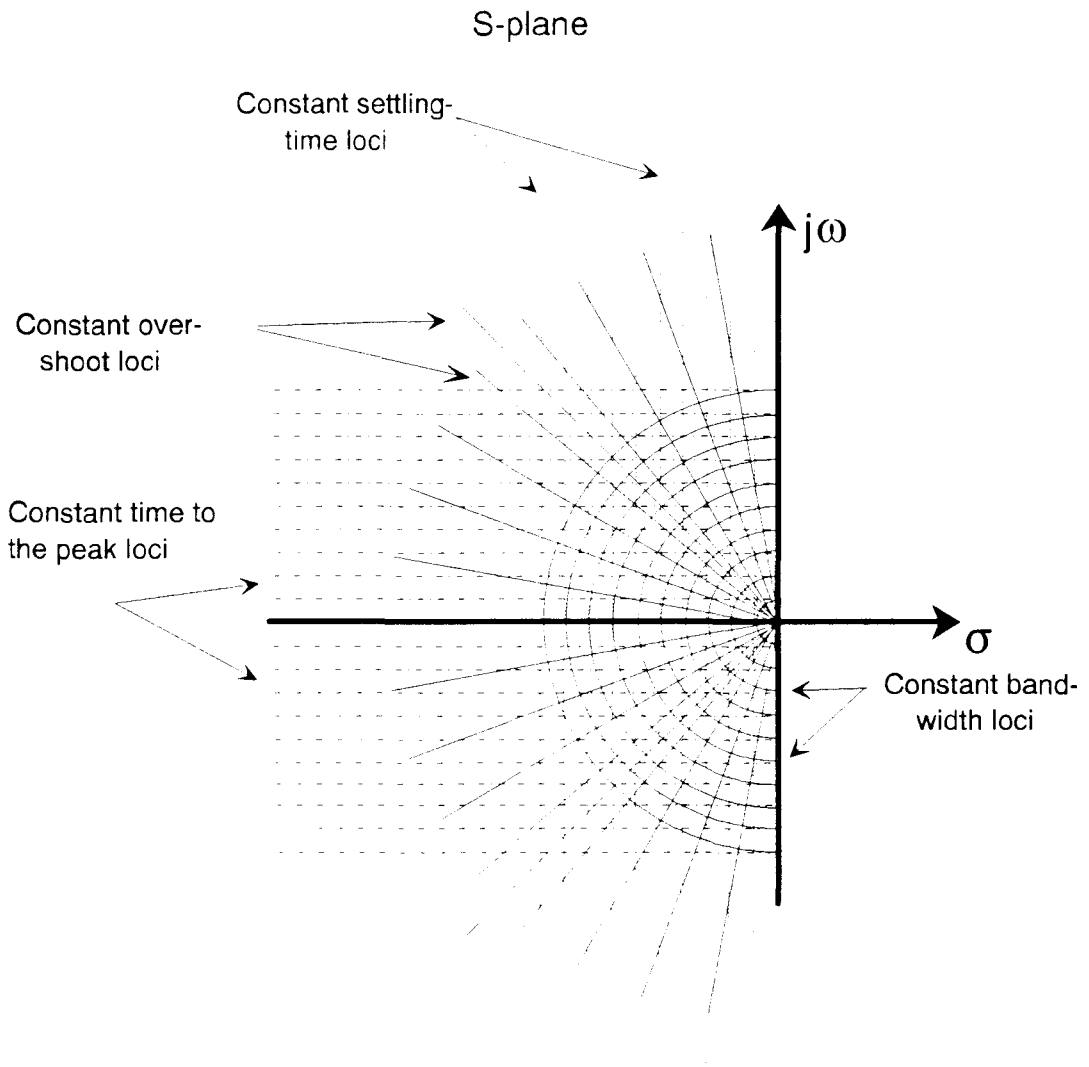


Figure 5.2 Performance Specifications Loci For Control System Approximated By A Pair Of Dominant Closed-Loop Poles

5.3 Preliminary Investigations Of The Plant Behaviour Using A Proportional-Term Controller

The preliminary investigations focus our attention to the controlling role of the loop gain as the simplest controller although it is clear from the outset that this type of controller can not satisfy the required steady-state error for a step input target (the resulting control system being type zero). The investigations, nevertheless, can reveal how far the loop-gain can modify the characteristics of the system in closed-loop mode. The CAD system produced root loci and closed-loop poles' sensitivity profiles as the gain is varied as shown in figures 5.3 - 5.6 which corresponds to the four cases of plant delays;

0.0, 0.05, 0.1 and 0.2 sec.

Concurrently the CAD system generated performance specifications loci in the S-plane which are super-imposed onto the gain control systems, also shown reproduced on a separate transparency associated with the control system of the zero-delay plant of figure 5.3. On the other hand, the sensitivity threshold requirements, although included in the screen display to assist the designer decision, are not shown on the graphs.

On closer examination of the composite screen displays (or the reproduced graphics), it is evident that the proportional-term controller cannot adequately satisfy all the specifications outlined in section 5.2.2., least of all the steady-state error which has been already anticipated earlier.

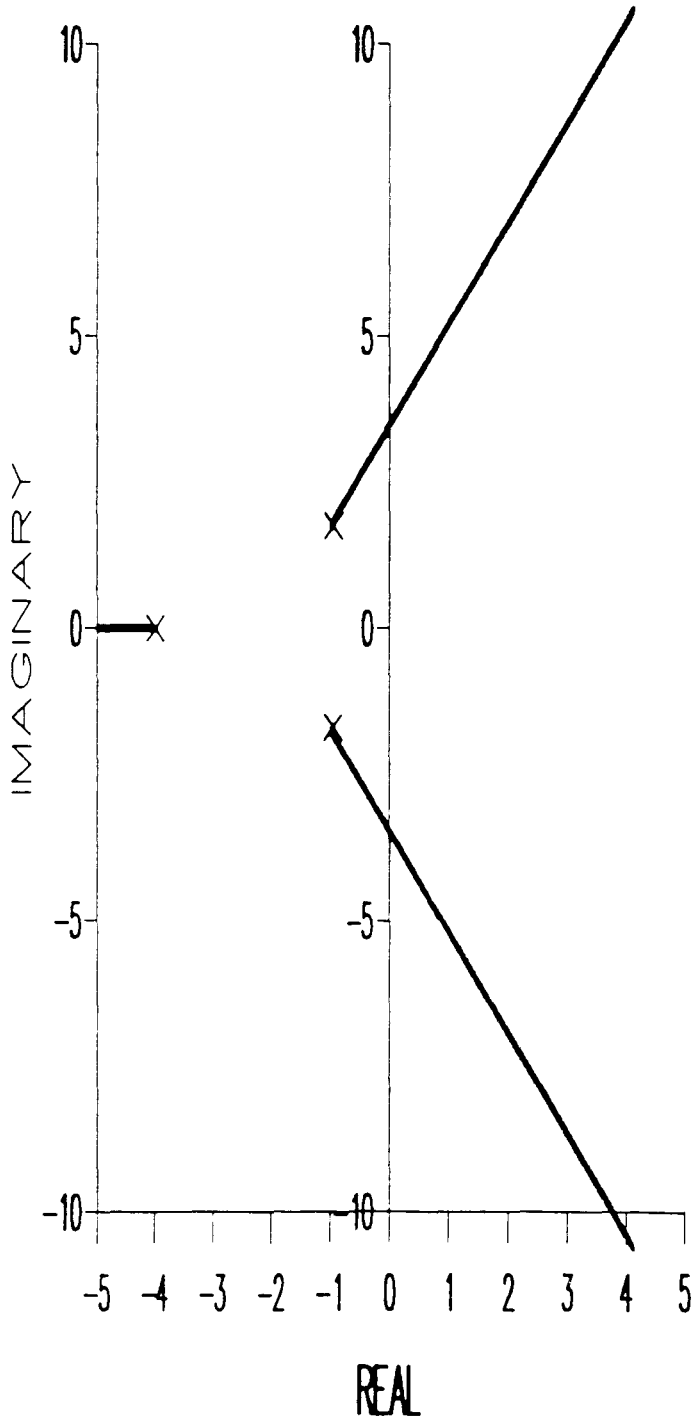


Figure 5.3A Root-Locus Diagram Of The Proportional-Term Control System For Plant $G_1(S)$ With Zero Delay (Performance Specifications Loci Are Superimposed)

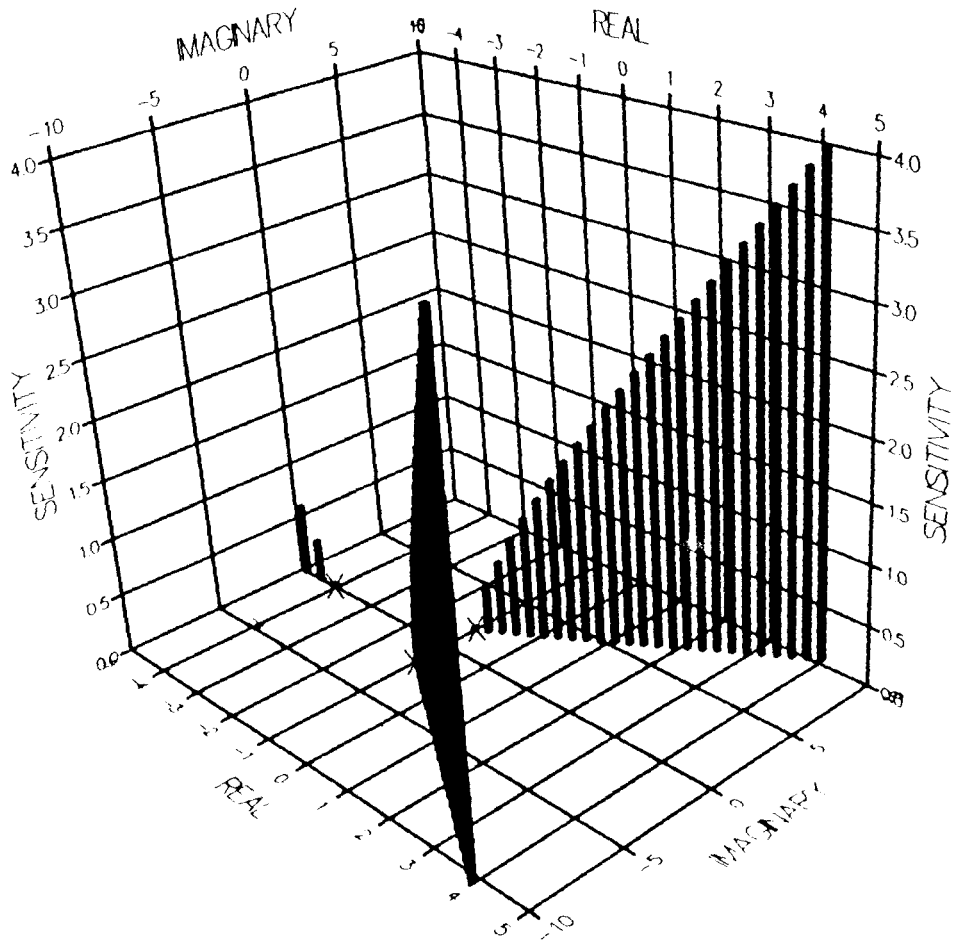


Figure 5.3B Root Sensitivity Profile Of The Proportional-Term Control System For Plant $G_1(S)$ With Zero s Delay

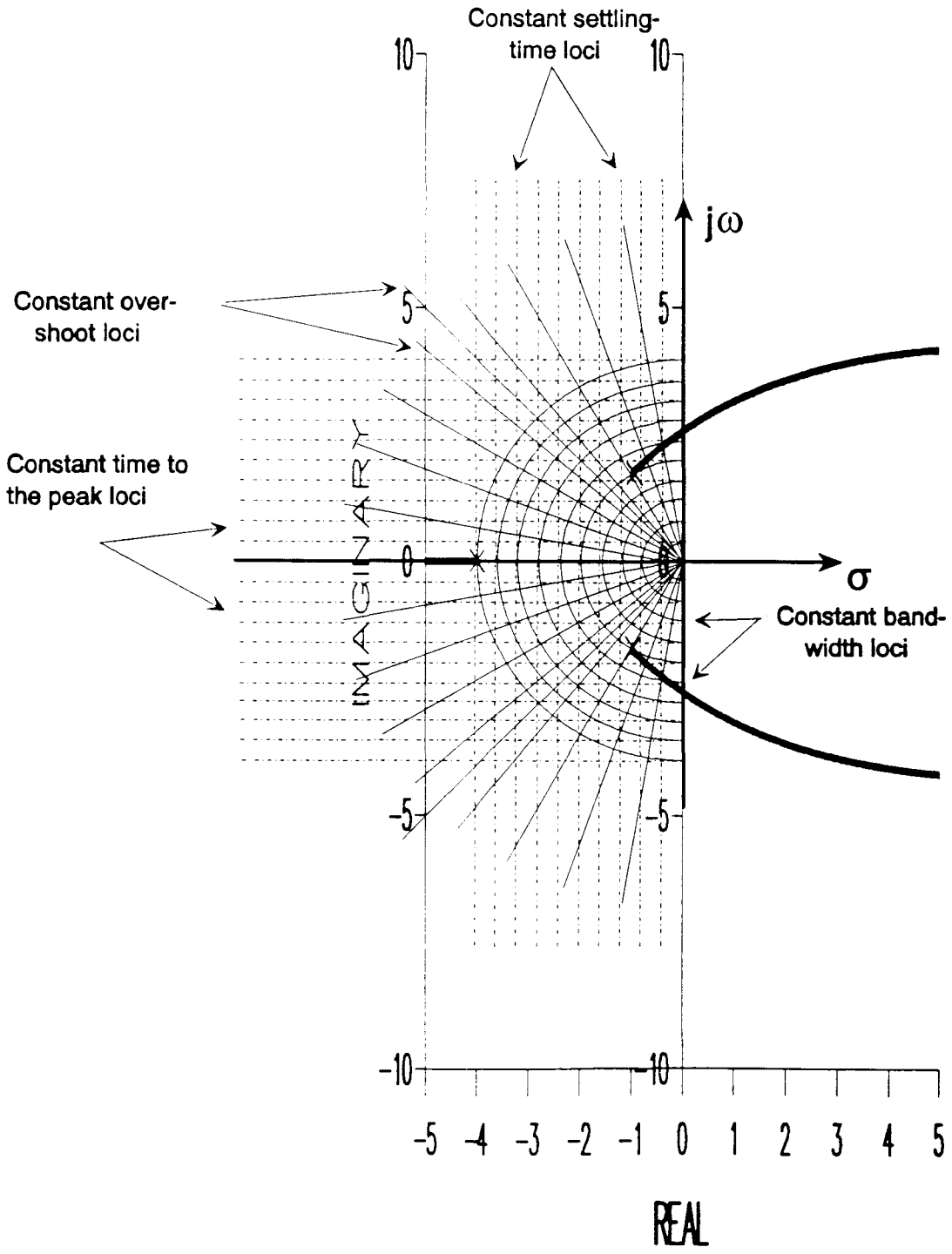


Figure 5.4A Root-Locus Diagram Of The Proportional-Term Control System For Plant $G_2(S)$ With 0.05 s Delay (Performance Specifications Loci Are Superimposed)

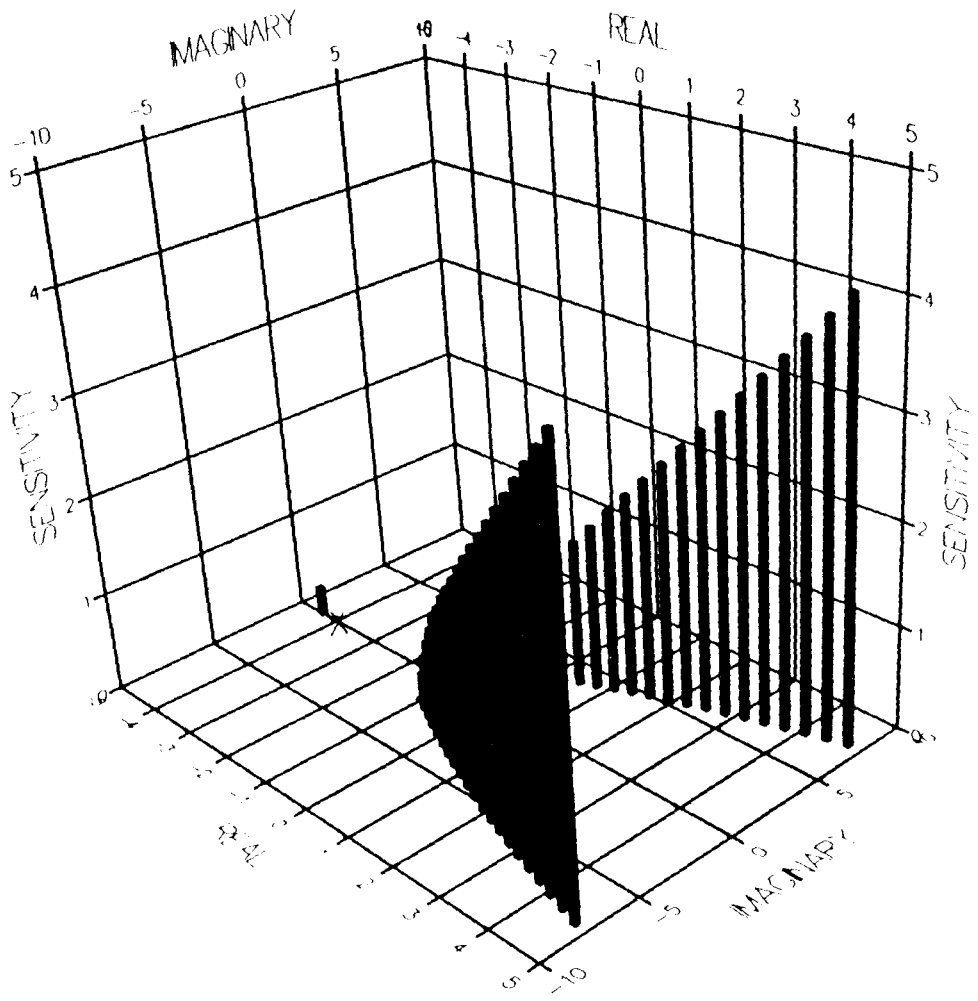


Figure 5.4B Root Sensitivity Profile Of The Proportional-Term Control System For Plant $G_2(S)$ With 0.05 s Delay

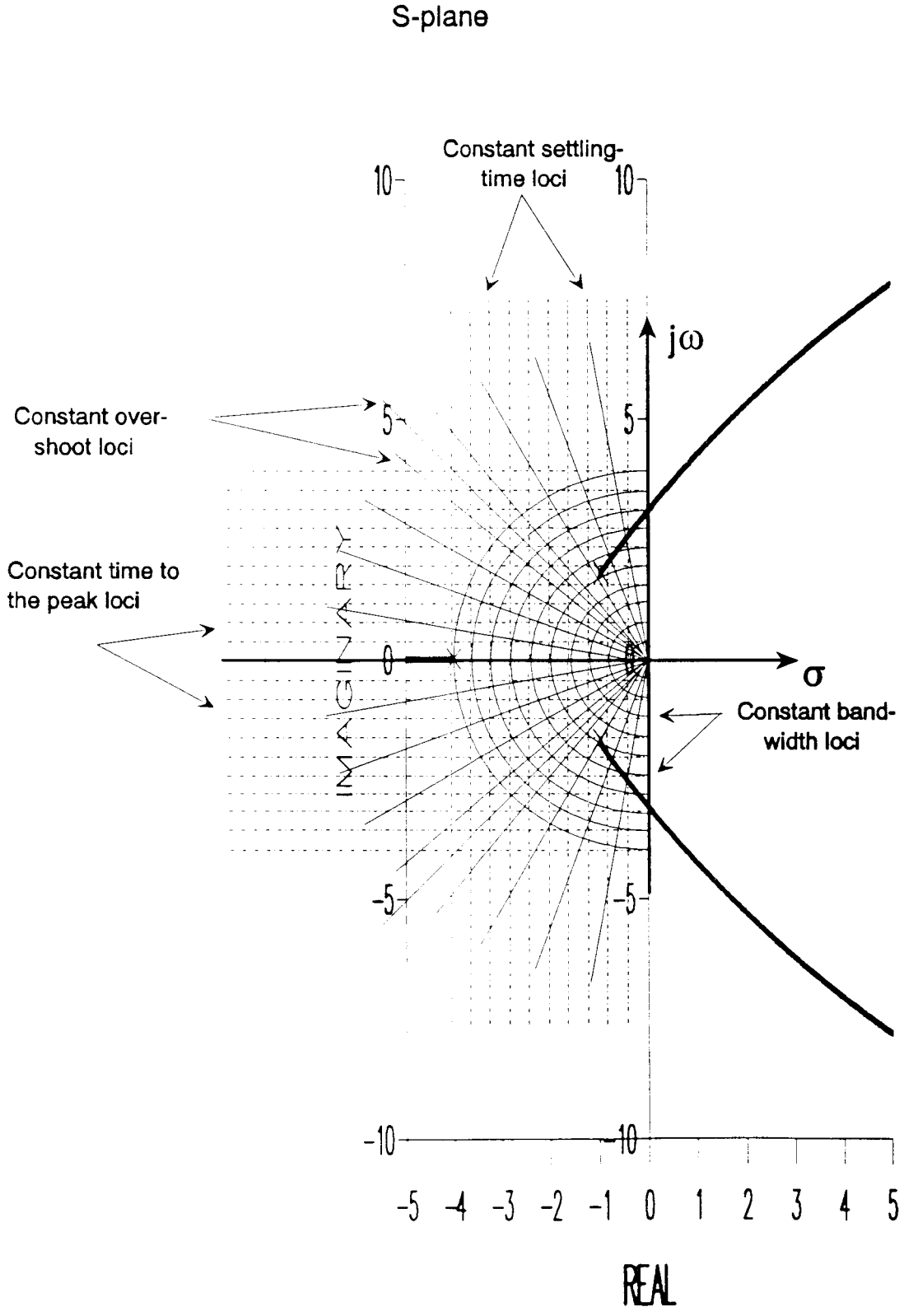


Figure 5.5A Root-Locus Diagram Of The Proportional-Term Control System For Plant $G_3(S)$ With 0.1 s Delay (Performance Specifications Loci Are Superimposed)

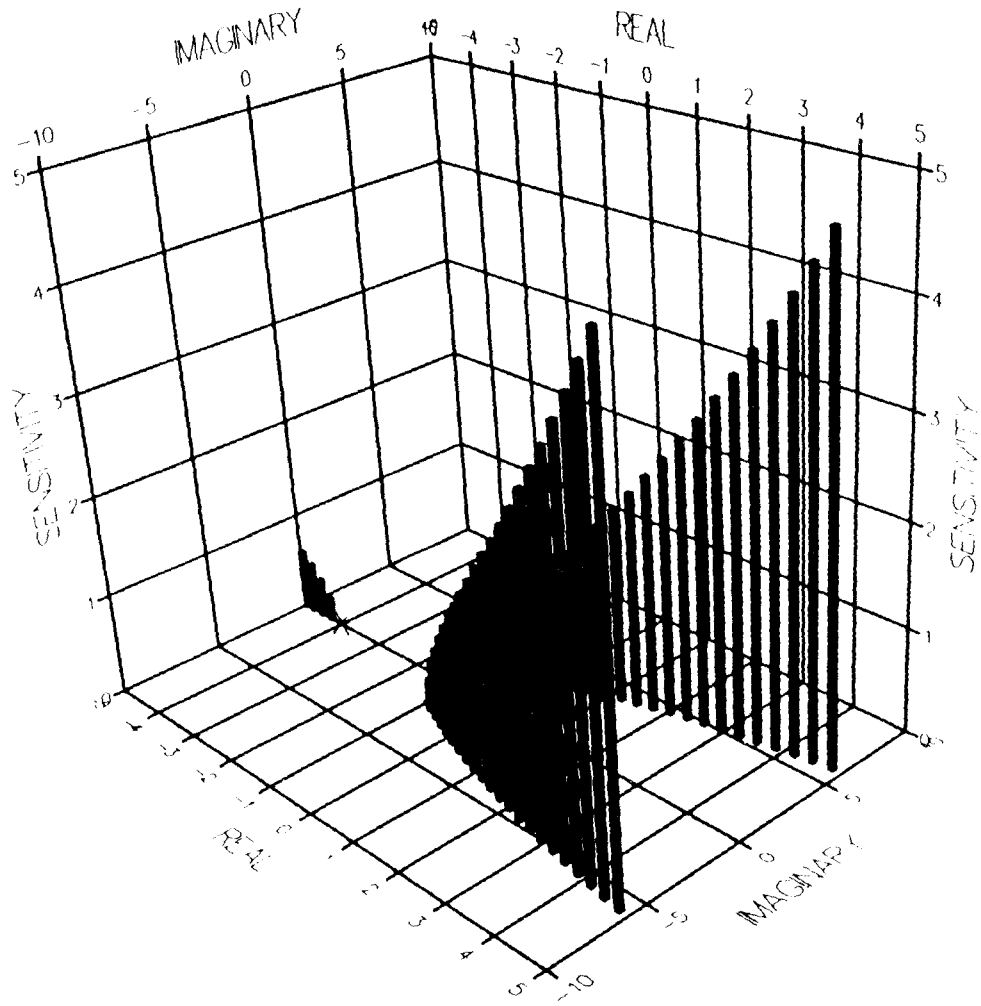


Figure 5.5B Root Sensitivity Profile Of The Proportional-Term Control System For Plant $G_3(S)$ With 0.1 s Delay

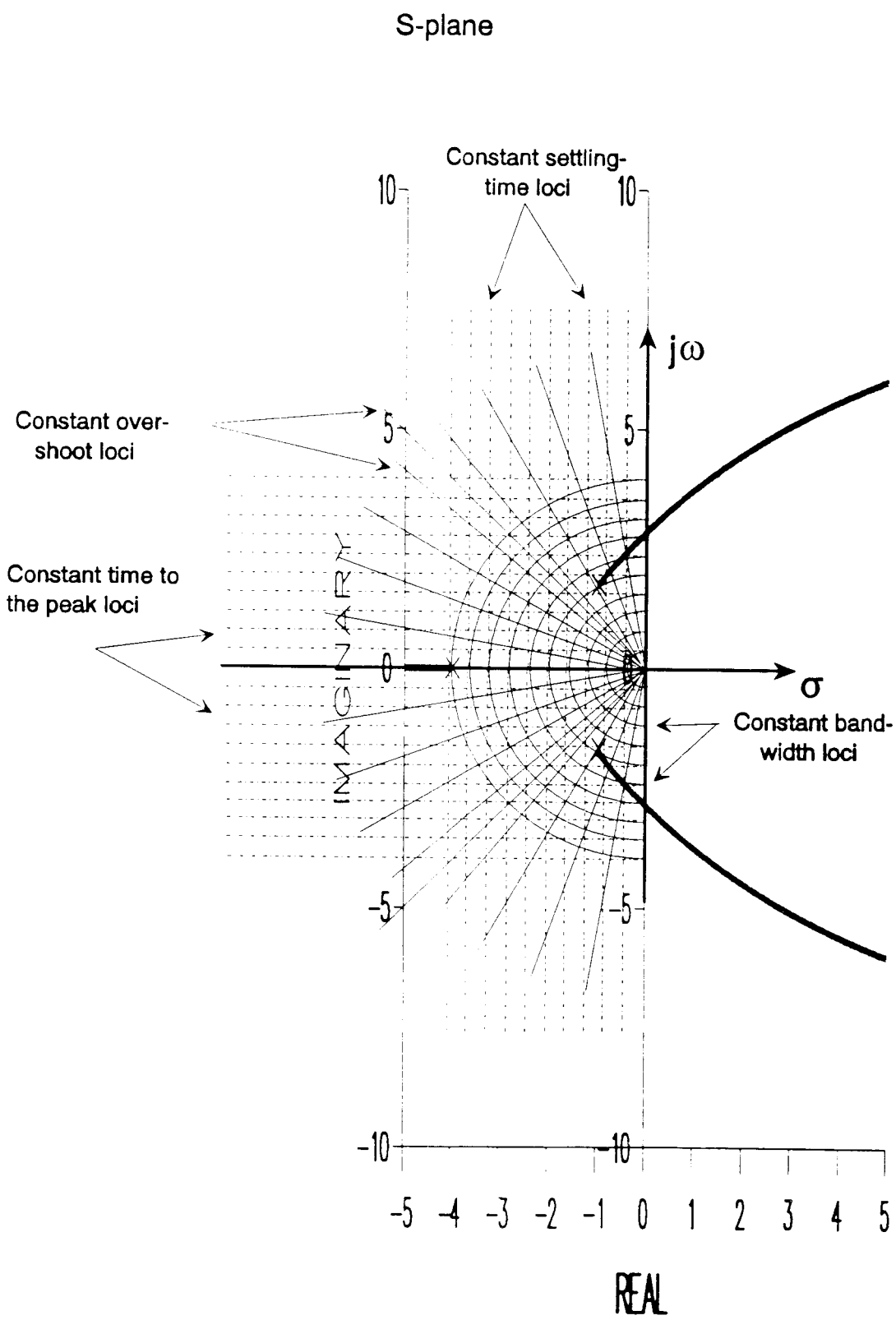


Figure 5.6A Root-Locus Diagram Of The Proportional-Term Control System For Plant $G_4(S)$ With 0.2 s Delay (Performance Specifications Loci Are Superimposed)

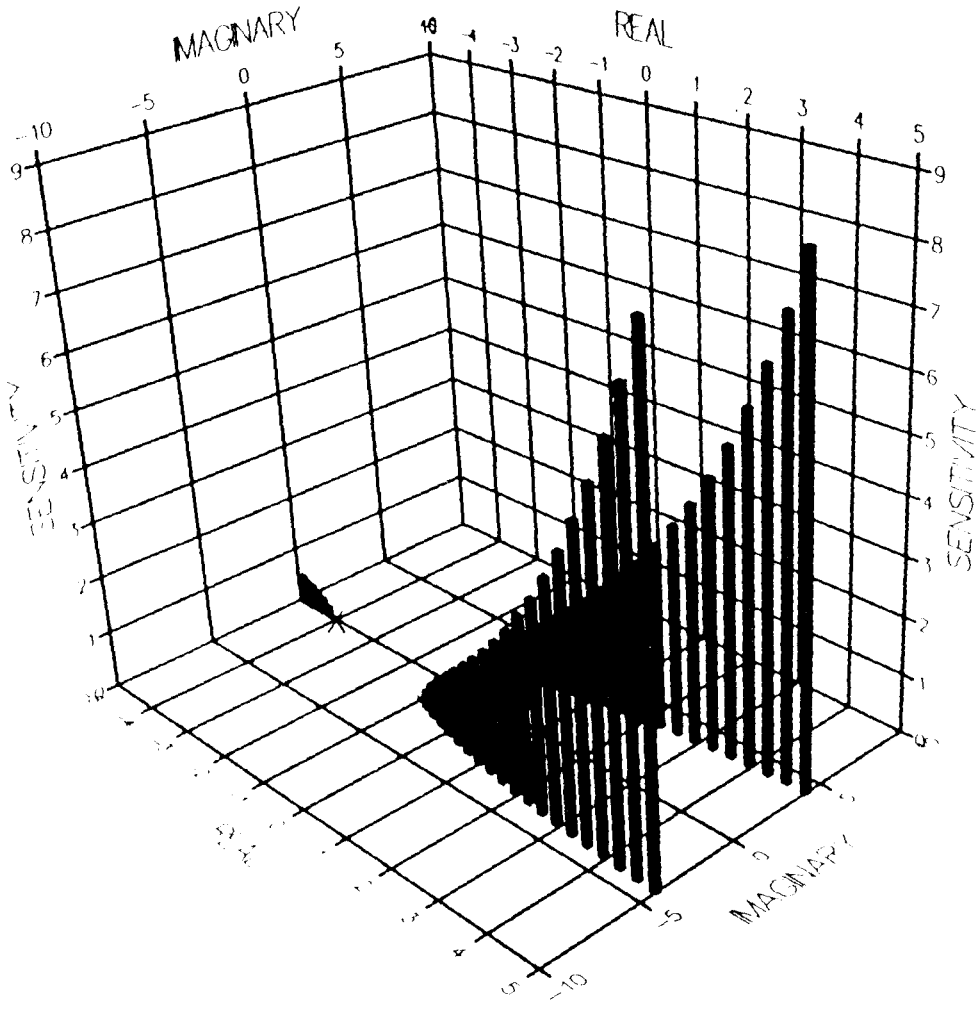


Figure 5.6B Root Sensitivity Profile Of The Proportional-Term Control System For Plant $G_4(S)$ With 0.2 s Delay

5.4 CAD Procedure For A Fixed- Configuration Optimum Controller Design

The inadequacies of the proportional -term controller, discussed in the pervious section, indicate the need for a controller which produces pole/zero configurations in addition to the gain control in order to effect wider and diverse modifications to the overall system performance. In this context, the poles of the resulting closed-loop transfer function "T(S)" determine the particular response modes that will be present, whereas the zeros of "T(S)" establish the relative weightings (or residues) of the individual mode functions. For example, moving a zero closer to a specific pole will reduce the relative contribution of the mode function corresponding to the pole, hence modifying the sensitivity of the relevant closed-loop pole (refer to chapter 4).

From the knowledge-base-sensitivity classification produced in chapter 4, a second-order configuration has been selected as the most appropriate for the autopilot control system. Locating one of the controller's poles at S=0 will satisfy the requirement of a zero steady-state error of the control system in response to a unit-step pitch target angle. In the meantime, the effect of the controller's second pole can be made insignificant by locating it far away on the real-axis of the S-domain; at a distance greater than or equal to 10 times the width of the active window within which the control system is expected to operate eventually. The controller's zeros, on the other hand, may be located as close as possible to the plant's dominant complex poles (at $S=-1.0 \pm 1.73$) so as to reduce their undesirable effect.

The CAD system has then been used interactively to view continuously and systematically the overall effect of moving the controller's zeros within a focused window of 0.5 radius surrounding each of the plant's complex poles excluding a zone of 0.05 radius centred on the same poles. These exclusion zones have been deliberately introduced to represent a degree of variability in the aircraft dynamics;

inevitably present in all real plants which render exact cancellation of the plant's poles practically impossible.

This search strategy has resulted in a fixed-configuration optimum controller design having the transfer function:

$$G_{CF}(S) = \frac{K_C\,(S^2 + 1.9\,S + 3.8954)}{S\,(S+40.0)} \qquad (5.12)$$

which is physically realisable by a PID (proportional plus integral plus derivative) controller.

Snap shots of the root-loci and the closed-loop poles' sensitivity profiles of the optimum autopilot design, together with the performance specifications loci superimposed, are shown in figures 5.7 - 5.10 for plants with delays of 0.0 - 0.2- sec. respectively .

Finally, the optimum loop gain "K" may be decided on the basis of the value which places the dominant complex poles of the closed-loop control system at their minimum sensitivity (i.e., maximum robustness), provided that all the other specifications have been also satisfied.

Since the loop gain $K = K_C\,K_P$ (5.13)

and the plant gain "K_P" is likely to vary in a real environment, then the optimum operation of the system can only be achieved if the controller gain K_C can be adaptively adjusted to maintain a loop gain value "K" which optimise the system's sensitivity.

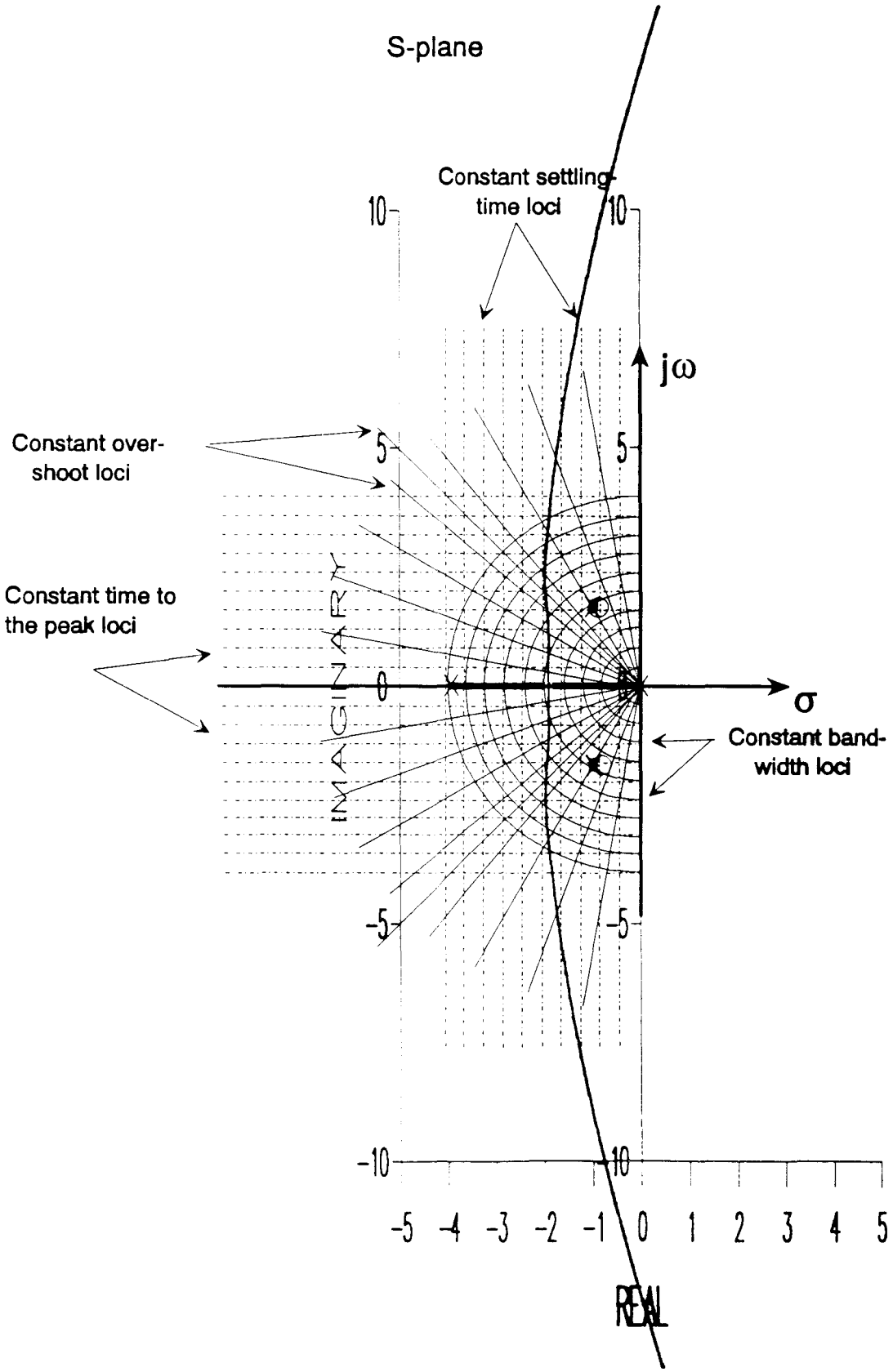


Figure 5.7A Root-Locus Diagram Of The Fixed-Configuration PID Control System For Plant $G_1(S)$ With Zero s Delay (Performance Specifications Loci Are Superimposed)

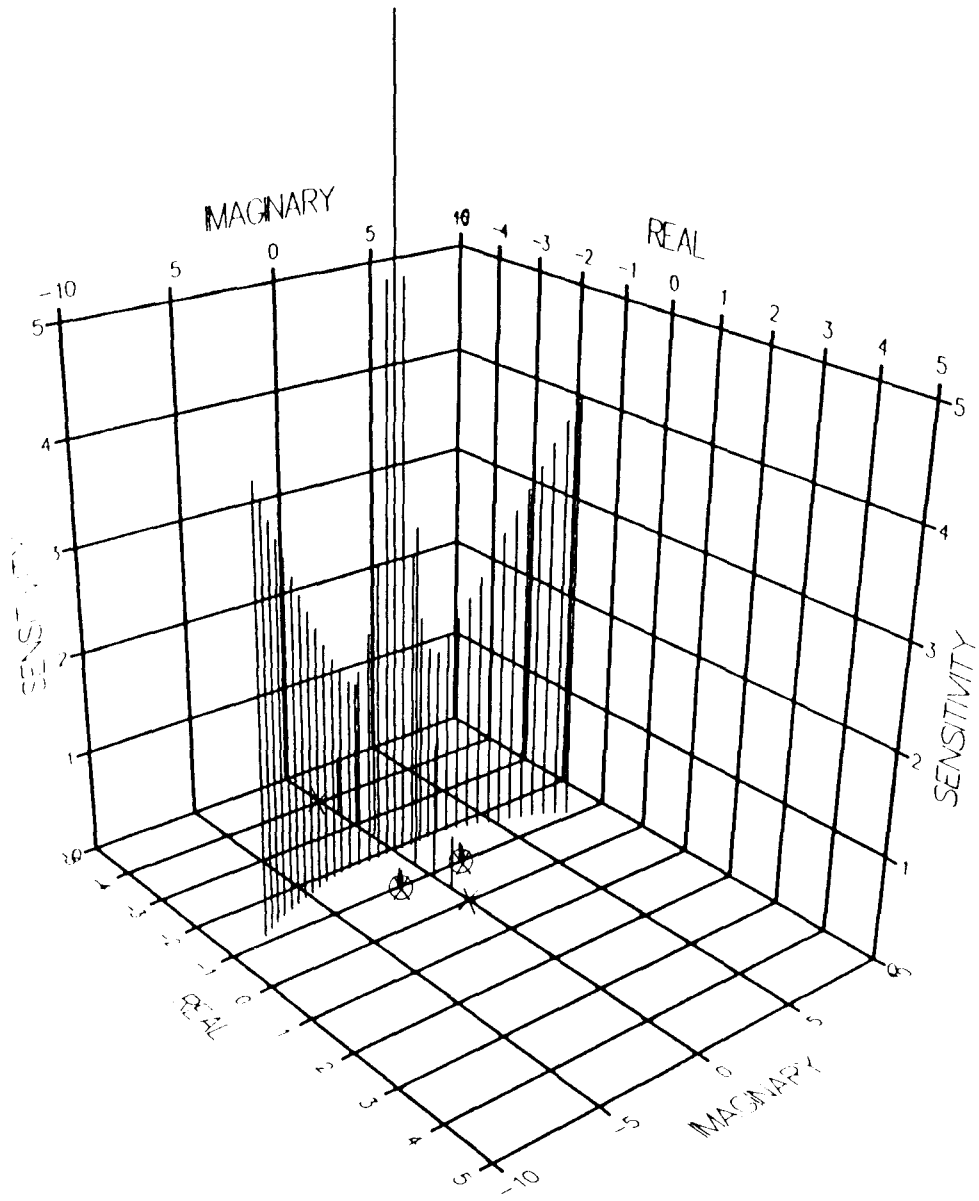


Figure 5.7B Root Sensitivity Profile Of The Fixed-Configuration PID Control System For Plant $G_1(S)$ With Zero s Delay

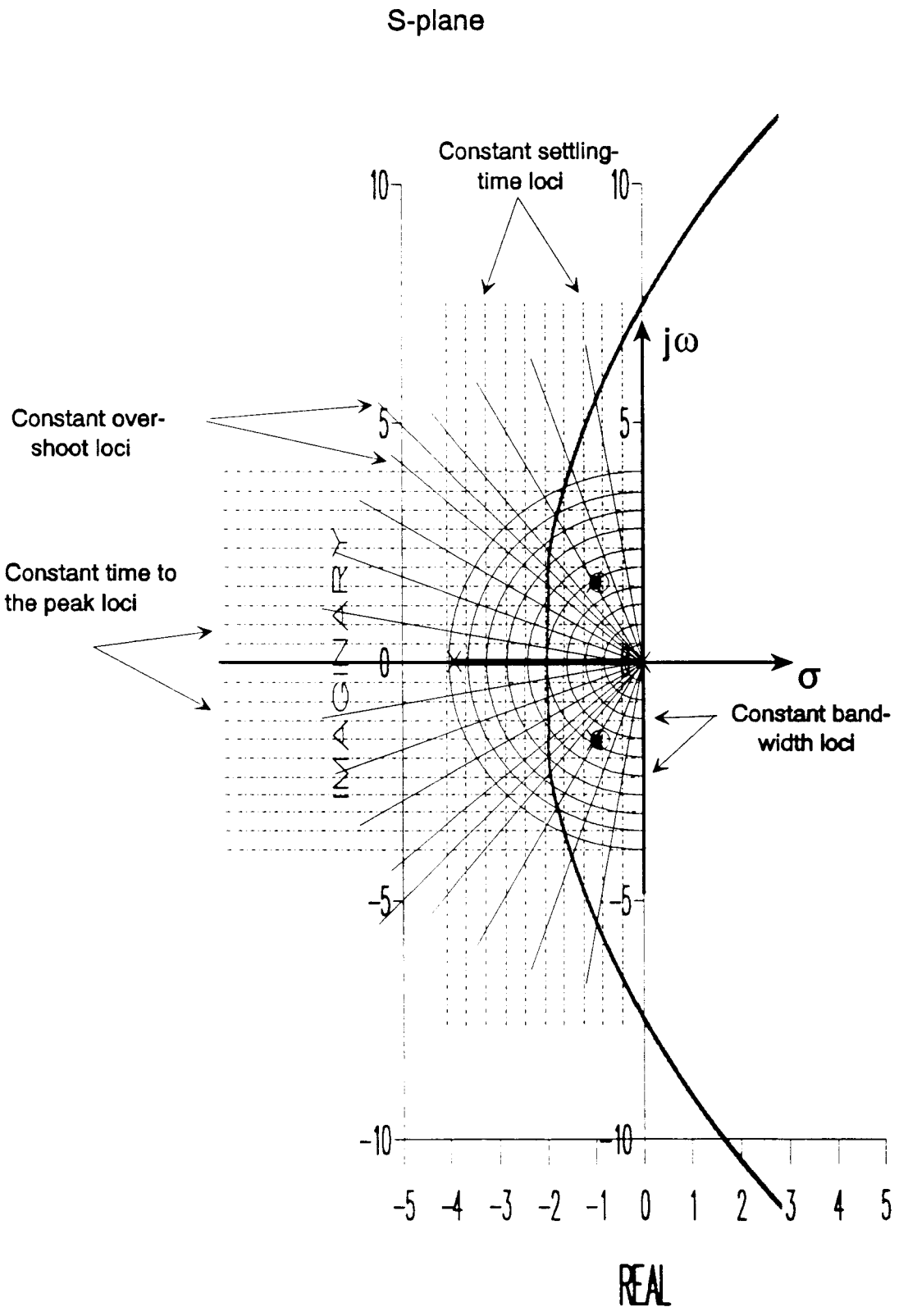


Figure 5.8A Root-Locus Diagram Of The Fixed-Configuration PID Control System For Plant $G_2(S)$ With 0.05 s Delay (Performance Specifications Loci Are Superimposed)

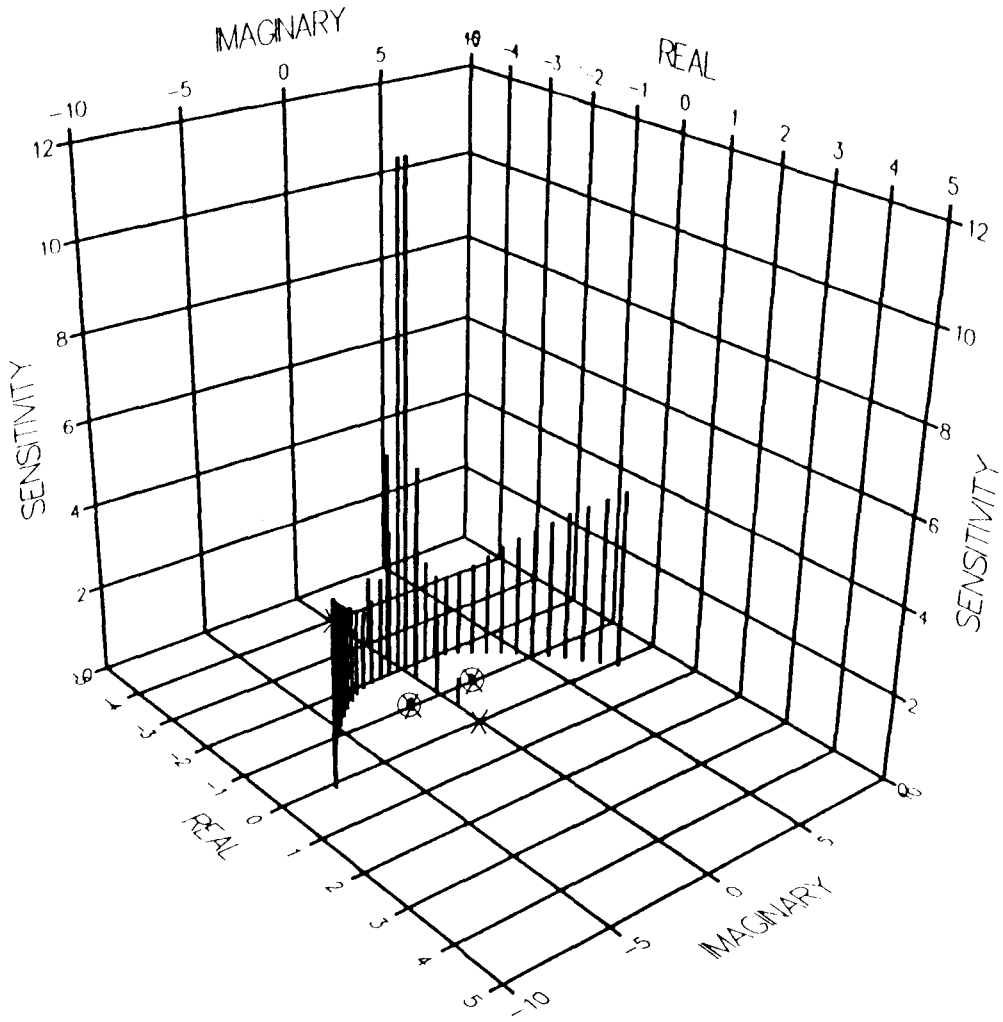


Figure 5.8B Root Sensitivity Profile Of The Fixed-Configuration PID Control System For Plant $G_2(S)$ With 0.05 s Delay

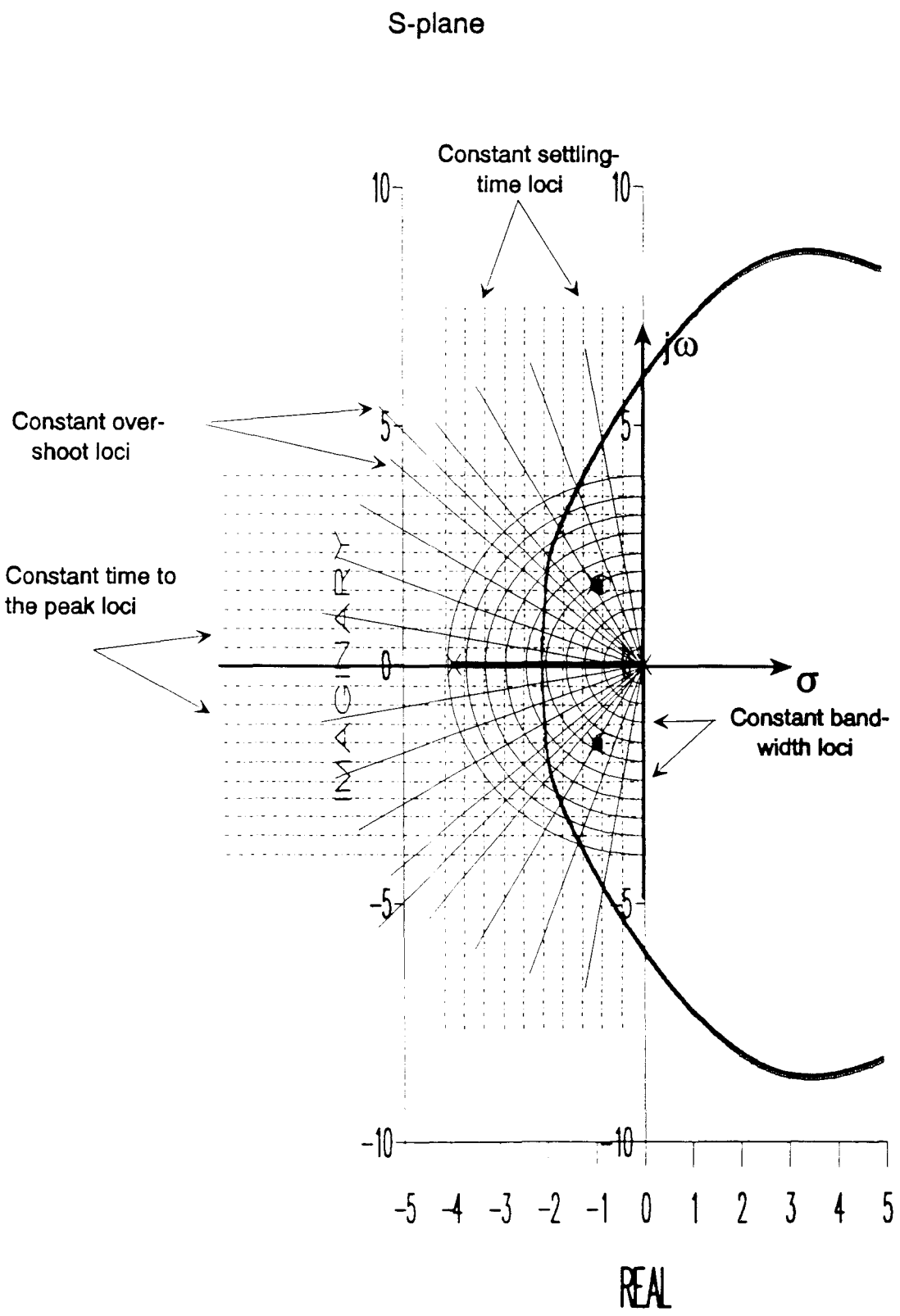


Figure 5.9A Root-Locus Diagram Of The Fixed-Configuration PID Control System For Plant $G_3(S)$ With 0.1 s Delay (Performance Specifications Loci Are Superimposed)

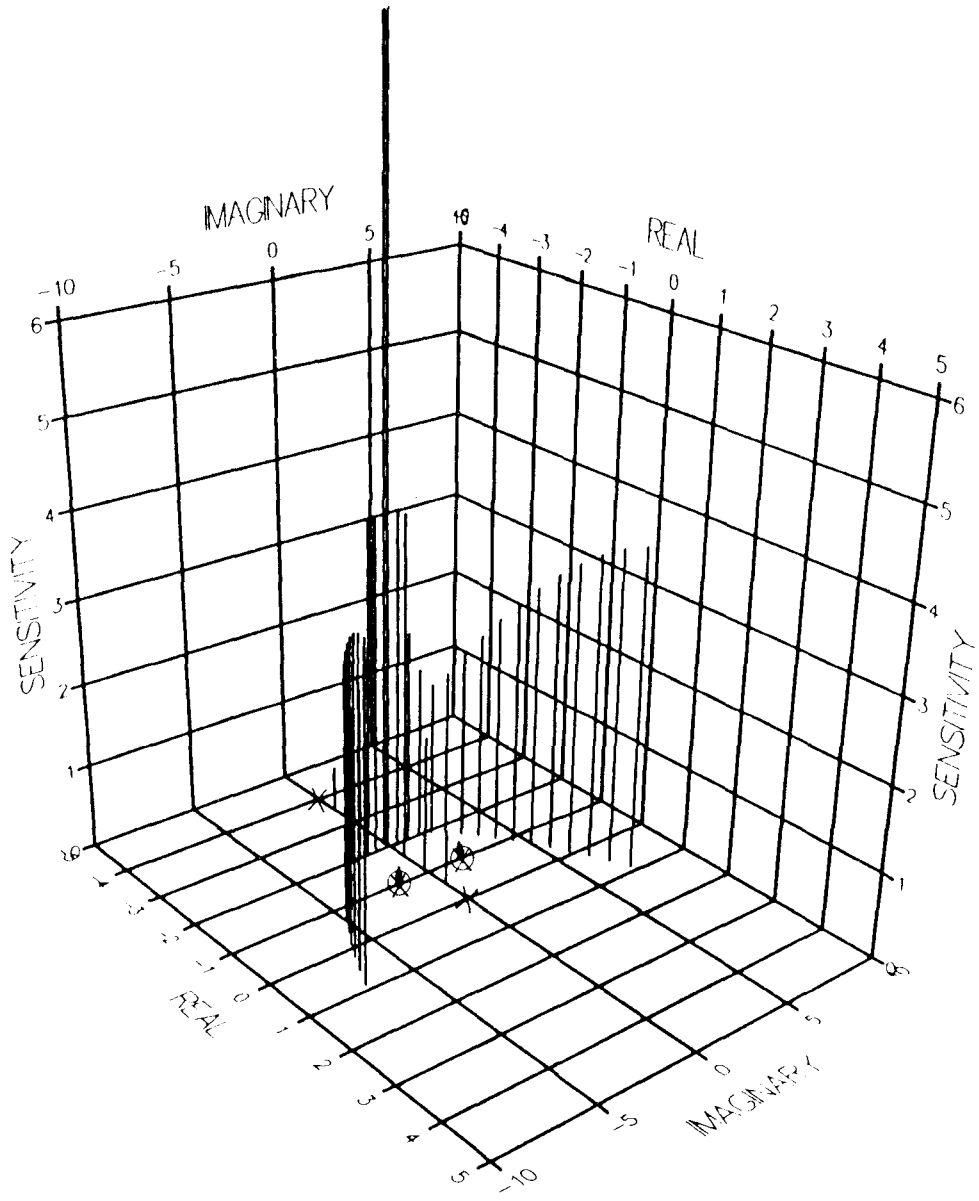


Figure 5.9B Root Sensitivity Profile Of The Fixed-Configuration PID Control System For Plant $G_3(S)$ With 0.1 s Delay

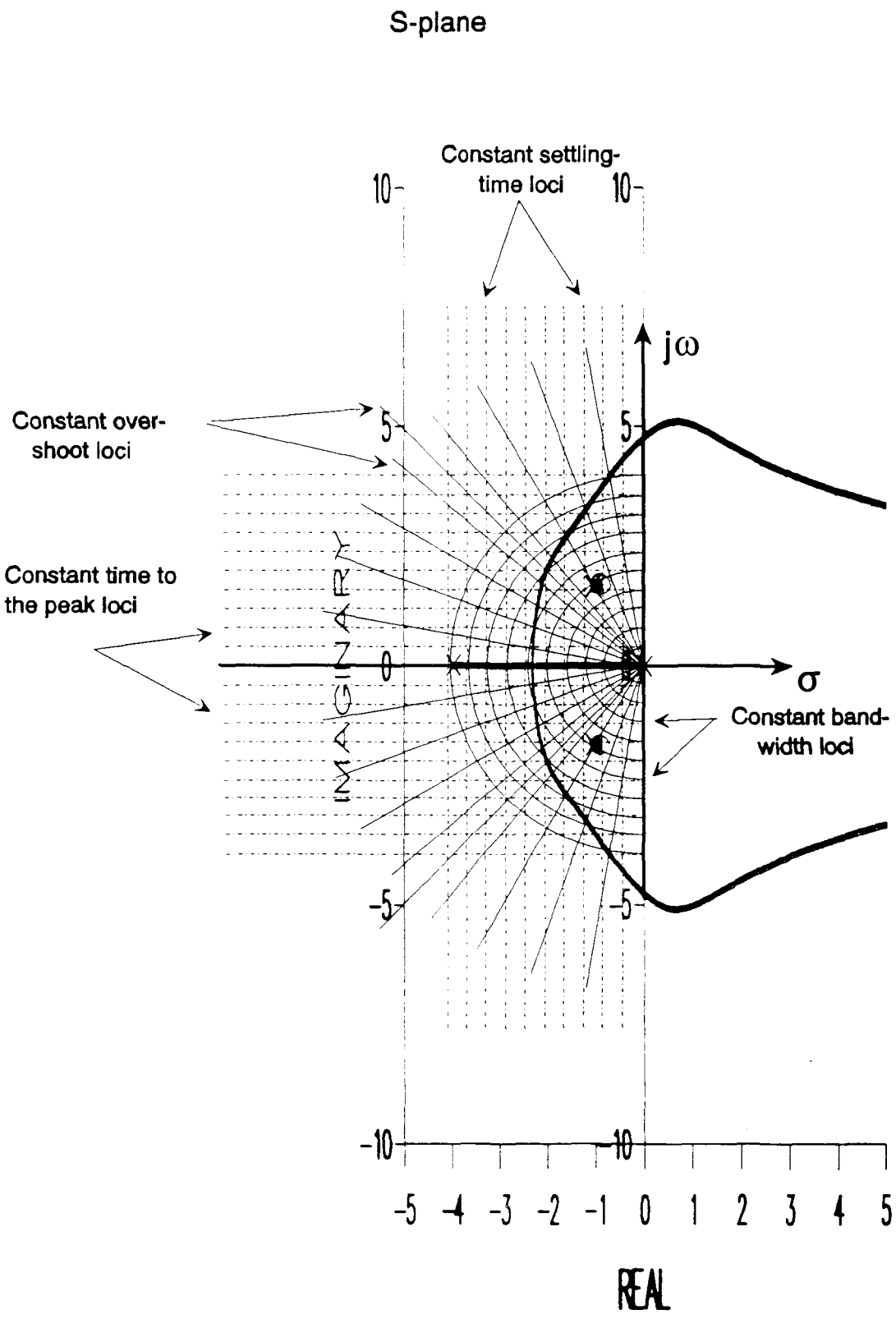


Figure 5.10A Root-Locus Diagram Of The Fixed-Configuration PID Control System For Plant $G_4(S)$ With 0.2 s Delay (Performance Specifications Loci Are Superimposed)

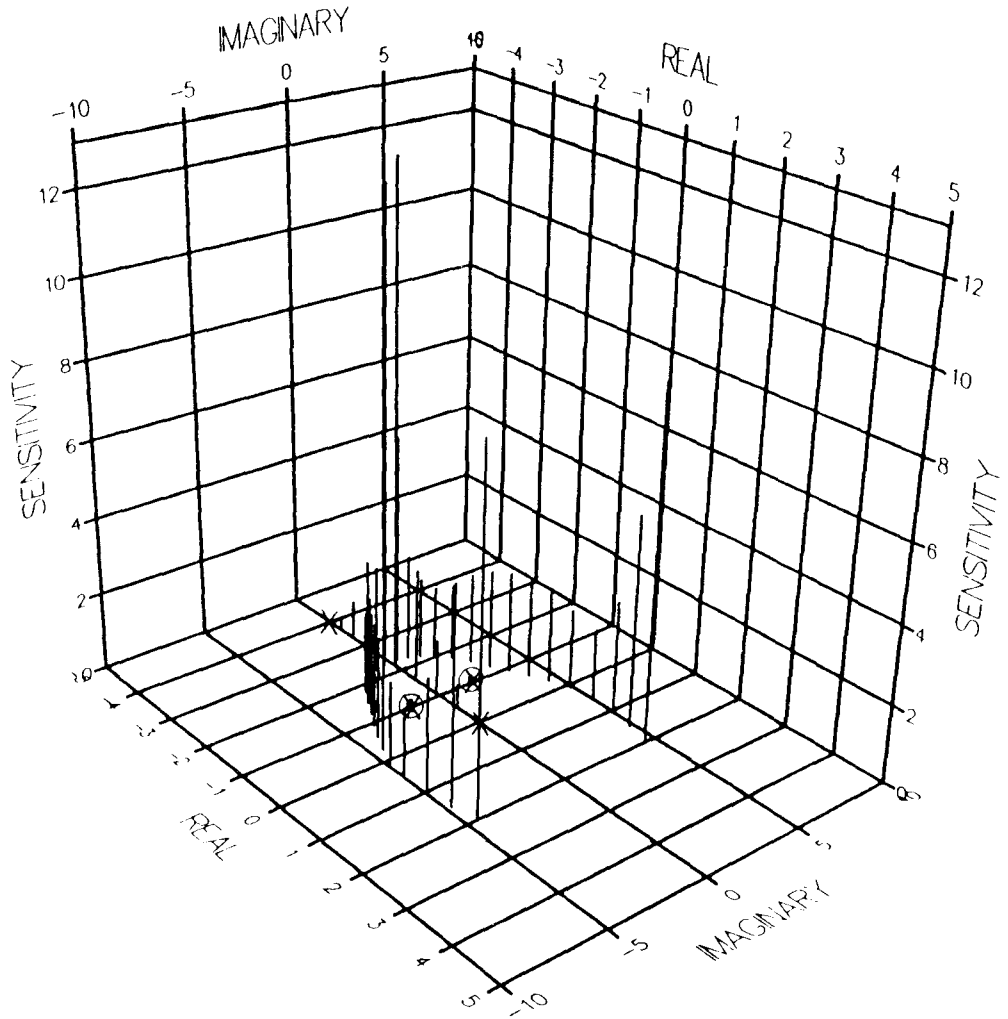


Figure 5.10B Root Sensitivity Profile Of The Fixed-Configuration PID Control System For Plant $G_4(S)$ With 0.2 s Delay

For a fixed-parameter controller, however, "K" is bound to reflect the variations in "K_P" once K_C is fixed. Under these conditions, the system will operate within a range of performance and sensitivities centred on its optimum. It follows, therefore, that the strategy of locating the controller's zeros should aim not only for optimising the system performance, but also for **minimising the operational zone of the system around its optimum for the range of variations of the plant parameters.** These underline the multi-faceted role of the developed CAD system and highlight its value in directing the designer iteratively to reach a compromise decision efficiently.

5.5 Evaluation Of The Results Of The Control System Design And Conclusions

Table 5.2 summarises the results (obtained from the CAD system) of the autopilot control system design using a fixed-configuration second-order controller as described by equation 5.12. The results are divided into four sets which are classified according to the plants delay (column 1). Each set is sub-divided into three sub-sets which refer to the loop-gain "K" adjustments:

i) "K" set at the system's minimum sensitivity of the dominant closed-loop poles,

ii) "K" set at approximately 5% below that of (i),

iii) "K" set at approximately 5% above that of (i)

to simulate plant gain variations of about $\pm$ 5%.

Examining the system's performance indices reveals that all the required specifications have been consistently satisfied even after allowing for relatively wide variations of both plant gain ($\pm$ 5%) and delay (in the range of 0.0 to 0.2 sec.). Concurrently and equally important, the designed controller has also narrowed considerably the

Table 5.2 Results Of The Fixed-Configuration PID, Control System's Design For A Band Of Sensitivities Of The Dominant Roots Centred On The Optimum Value (Figures Rounded To Four Decimal Places).

Loop-Gain Settings*	Plant's Delay (sec.)	Sensitivity Of The Dominant Closed-Loop Poles	Bandwidth (rad. s^{-1})	Loop Gain ($K_P K_C$) (rad. s^{-1})	Attenuation Required Outside The Loop For Unity d.c. Gain	Velocity Error Constant K_V (rad. s^{-1})	Steady-State Error For A Unit-Ramp Input	Time-Domain Performance For A Unit-Step Target Input		
								Percent Overshoot	Settling Time (sec.)	Rise Time (sec.)
Lower Threshold		1.8939	2.8702	306.7360	1.0000	1.8670	0.5356	4.0070	2..3636	0.8028
Optimum	0.0	1.8889	2.9474	324.2554	1.0000	1.9736	0.5067	4.7480	2.3636	0.7500
Upper Threshold		1.8947	3.0253	342.7698	1.0000	1.9767	0.5059	5.4790	2.3656	0.6972
Lower Threshold		1.8860	2.8716	339.4048	0.8952	2.0658	0.4841	4.0430	2.3665	0.8028
Optimum	0.05	1.8809	2.9411	358.8757	0.8898	2.1483	0.4578	4.6810	2.3802	0.7500
Upper Threshold		1.8879	3.0109	379.4510	0.8849	2.3096	0.4330	5.6380	2.3970	0.7077
Lower Threshold		1.8780	2.8442	370.0994	0.8808	2.2526	0.4439	3.4040	2.3529	0.8239
Optimum	0.1	1.8716	2.9045	391.4342	0.7902	2.3825	0.4197	4.3620	2.3790	0.7711
Upper Threshold		1.8800	2.9647	413.9847	0.7804	2.5197	0.3969	5.3190	2.4097	0.7289
Lower Threshold		1.8741	2.7727	430.2689	0.6563	2.6189	0.3818	2.1280	2.2579	0.9401
Optimum	0.2	1.8672	2.8122	455.1312	0.6342	2.7702	0.3610	2.4470	2.3004	0.8768
Upper Threshold		1.8783	2.8810	481.4018	0.6141	2.9301	0.3413	3.4040	2.3495	0.8240

* Refers To The Upper & The Lower Boundaries Of The Loop-Gain Band (Of $\pm$5%) Centred On Its Value At The Minimum Sensitivity Of The Dominant Closed-Loop Poles.

variations of all the indices optimally; resulting in a fairly robust control system. It is interesting to note the following observations as the plant's delay increases:

* The steady- state error for a unit pitch-rate target consistently decreases (see column 7 of table 5.2),
* The bandwidth deteriorates (see column 3 of table 5.2),
* Rise-time deteriorates (see column 10 of table 5.2),
* Percent overshoot decreases (see column 8 of table 5.2),
* Settling time shows insignificant change (see column 9 of table 5.2),
* An adjustable attenuation unit is required to be incorporated outside the designed control loop in order to maintain unity d.c. gain response when the plant has a finite delay. The amount of this attenuation increases consistently with delay increases (see column 5 of table 5.2).

To obtain a quantitative measure or measures of the system's robustness and to confirm its time-domain performance indices, the designed control system has been simulated (using Program CC version 4, see reference [57]), and a series of d.c. gain normalised transient responses were obtained for a unit-step pitch target angle. Figures 5.11 to 5.14 show four sets of these transient responses which correspond to plant's delays of 0.0,0.05, 0.1 and 0.2 seconds respectively. Each set consists of three responses recorded according to the loop-gain range which correspond to the plant's gain variations of $\pm$5.0% around a nominal value.

These graphs, apart from corroborating the results of the time-domain performance indices predicted by our CAD system (as given in columns 8,9 and 10 of table 5.2), they also show fairly narrow variations relative to each other. This indicates a low sensitivity of the control system to plant parameter variations (in terms of gain and delay). Table 5.3 gives the mean deviation (mean of the absolute differences) and the mean variance (mean square of the differences) of each set of responses(for a specific

plant delay) relative to the relevant response when the loop-gain is set at the system's minimum sensitivity of the dominant closed-loop poles.

These measures give clear indications to satisfactory system's robustness for plant's gain variations in the range of $\pm 5.0\%$ about a nominal value, since the mean deviation and the mean variance do not exceed 0.0061 and 0.00013 respectively (which occurred at plant delay of 0.1 seconds and at a loop gain of 5% above its optimum value, shown highlighted).

Grouping the same transient responses differently in order to derive system's sensitivity measures for plant's delay variations, results in figures 5.15 to 5.17 where the delayed responses have been deliberately compensated for graphically(by advancing the relevant graphs horizontally by the amount of the plant's delay for convenient comparisons). Again the mean deviation and the mean variance have been calculated

For each set relative to the relevant response when the plant has a zero delay, and the results are given in table 5.4. These results provide a further support to the quality assurance of the control system behaviour, designed using the developed sensitivity-based CAD system, where the mean deviation and the mean variance do not exceed 0.0163 and 0.00091 respectively (which occurred at a loop gain of 5% above its optimum value and at a plant delay of 0.2 seconds, shown highlighted).

Table 5.3 Quality Assurance Measures Of The Fixed-Configuration PID. Control System In Terms Of Transient Response Sensitivity To Plant Gain Variations Of ± 5%, For Delays Of 0.0, 0.05, 0.1, And 0.2 Sec. Respectively.

Loop-Gain Settings*	Plant Delay (sec.)	Transient Response Sensitivity To Plant Gain Variations Of ± 5% Relative To Its Optimum Value.	
		Mean Deviation	Mean Variance
Lower Threshold	0.0	0.005849	0.000123
Upper Threshold		0.005700	0.000120
Lower Threshold	0.05	0.005969	0.000125
Upper Threshold		0.005900	0.000125
Lower Threshold	0.1	0.006025	0.000122
Upper Threshold		**0.006060**	**0.000127**
Lower Threshold	0.2	0.005374	0.000087
Upper Threshold		0.005692	0.000100

* Refers To The Upper & The Lower Boundaries Of The Loop-Gain Band (Of ± 5%) Centred On Its Value At The Minimum Sensitivity Of The Dominant Closed-Loop Poles.

Table 5.4 Quality Assurance Measures Of The Fixed-Configuration PID. Control System In Terms Of Transient Response Sensitivity To Plant Delay Variations Of 0.0- 0.2 Sec., For a Gain Band of ± 5%.

Loop-Gain Settings*	Plant Delay (sec.)	Transient Response Sensitivity To Plant Delay Variations Relative To That Of Zero Delay.	
		Mean Deviation	Mean Variance
Minimum Sensitivity	0.05	0.000695	0.000001
	0.1	0.003987	0.000052
	0.2	0.016015	0.000840
Lower Threshold	0.05	0.000606	0.000001
	0.1	0.003999	0.000050
	0.2	0.015480	0.000748
Upper Threshold	0.05	0.000886	0.000002
	0.1	0.003975	0.000053
	0.2	**0.016259**	**0.000907**

* Refers To The Upper & The Lower Boundaries Of The Loop-Gain Band (Of ± 5%) Centred On Its Value At The Minimum Sensitivity Of The Dominant Closed-Loop Poles.

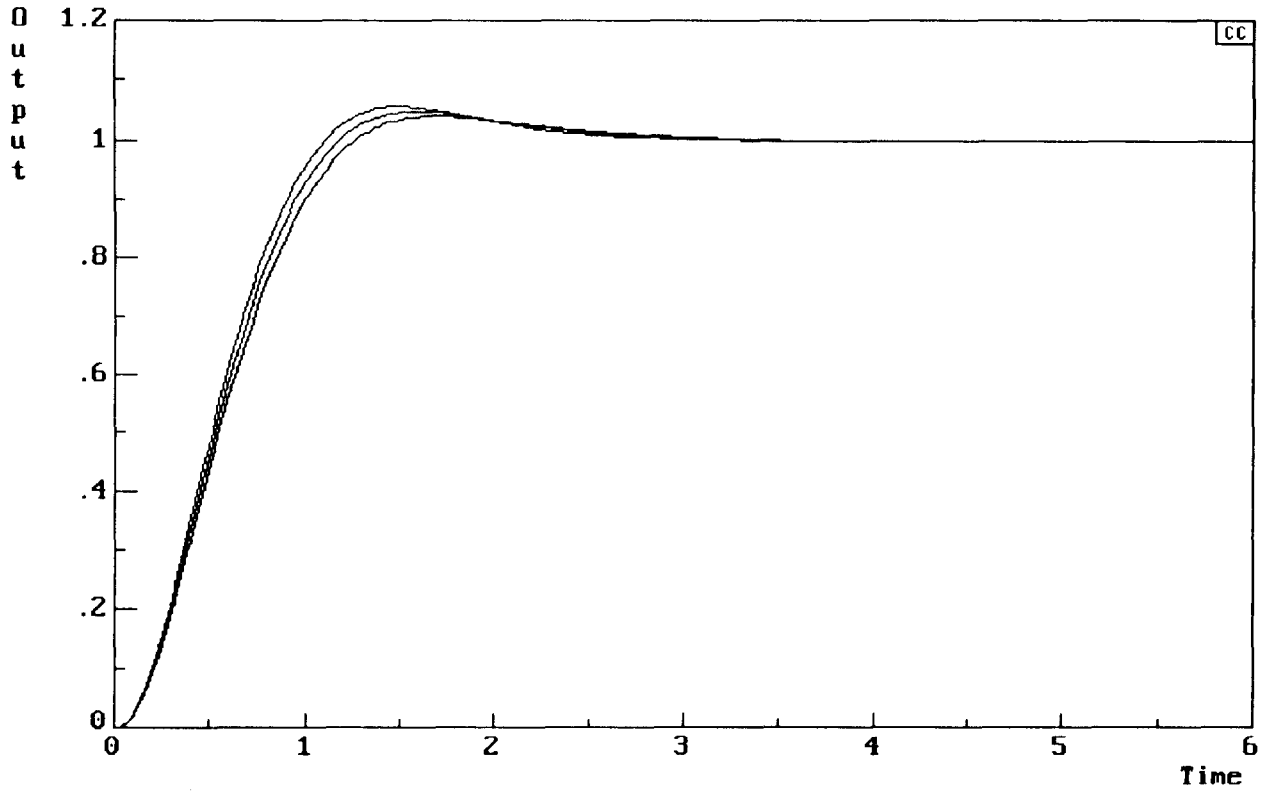


Figure 5.11 Family Of Transient Responses Of The Fixed-Configuration PID Control System For Plant $G_1(S)$ With Zero Sec. Delay, And For A Loop-Gain Band Of $\pm$ 5% Relative To Its Value At The Minimum Sensitivity Of The Dominant Closed-Loop Poles.

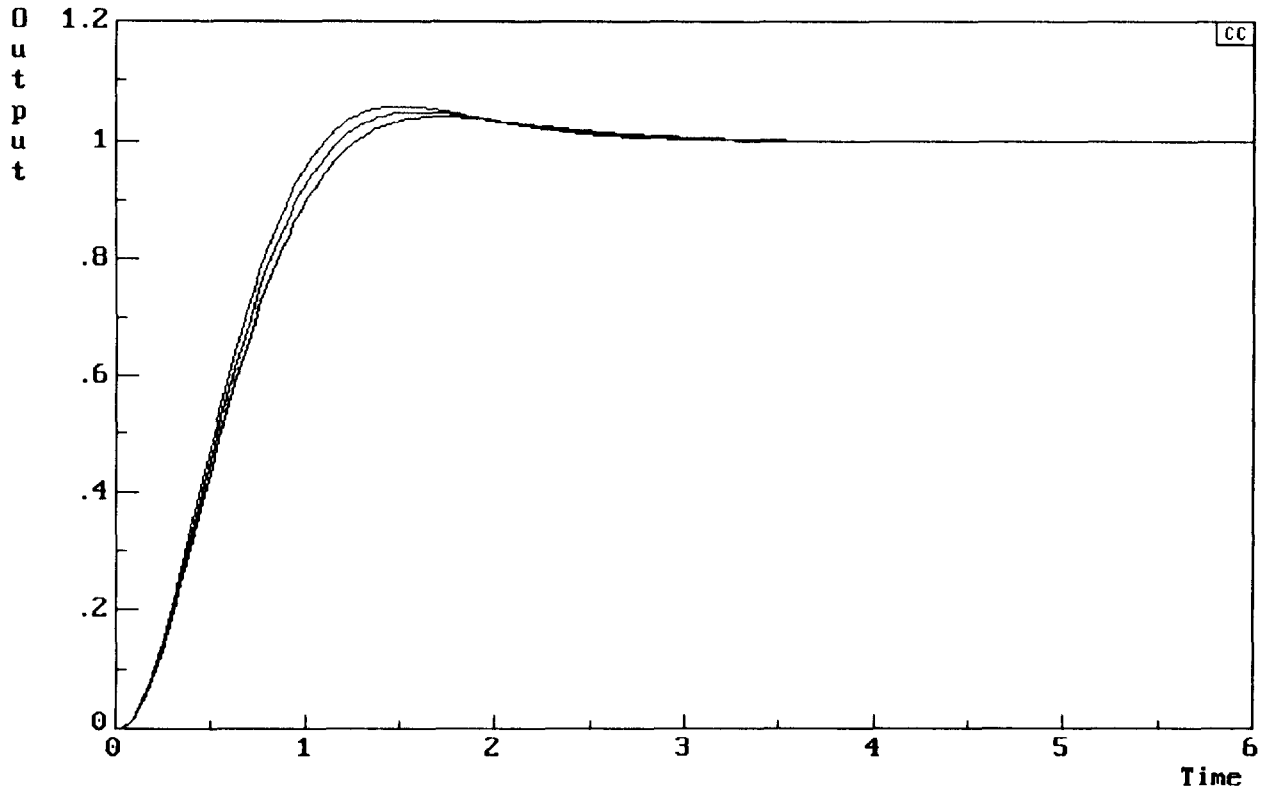


Figure 5.12 Family Of Transient Responses Of The Fixed-Configuration PID Control System For Plant $G_2(S)$ With 0.05 Sec. Delay, And For A Loop-Gain Band Of $\pm$ 5% Relative To Its Value At The Minimum Sensitivity Of The Dominant Closed-Loop Poles.

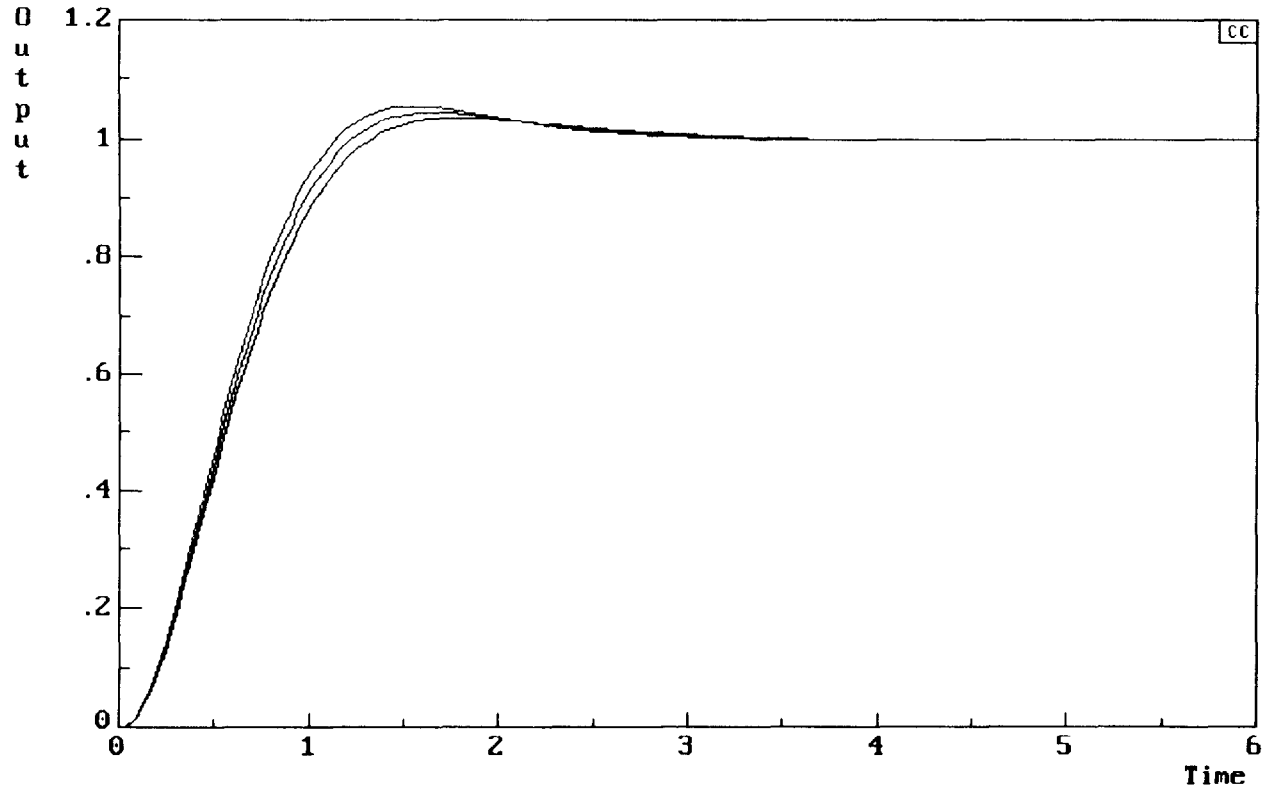


Figure 5.13 Family Of Transient Responses Of The Fixed-Configuration PID Control System For Plant $G_3(S)$ With 0.1 Sec. Delay, And For A Loop-Gain Band Of $\pm$ 5% Relative To Its Value At The Minimum Sensitivity Of The Dominant Closed-Loop Poles.

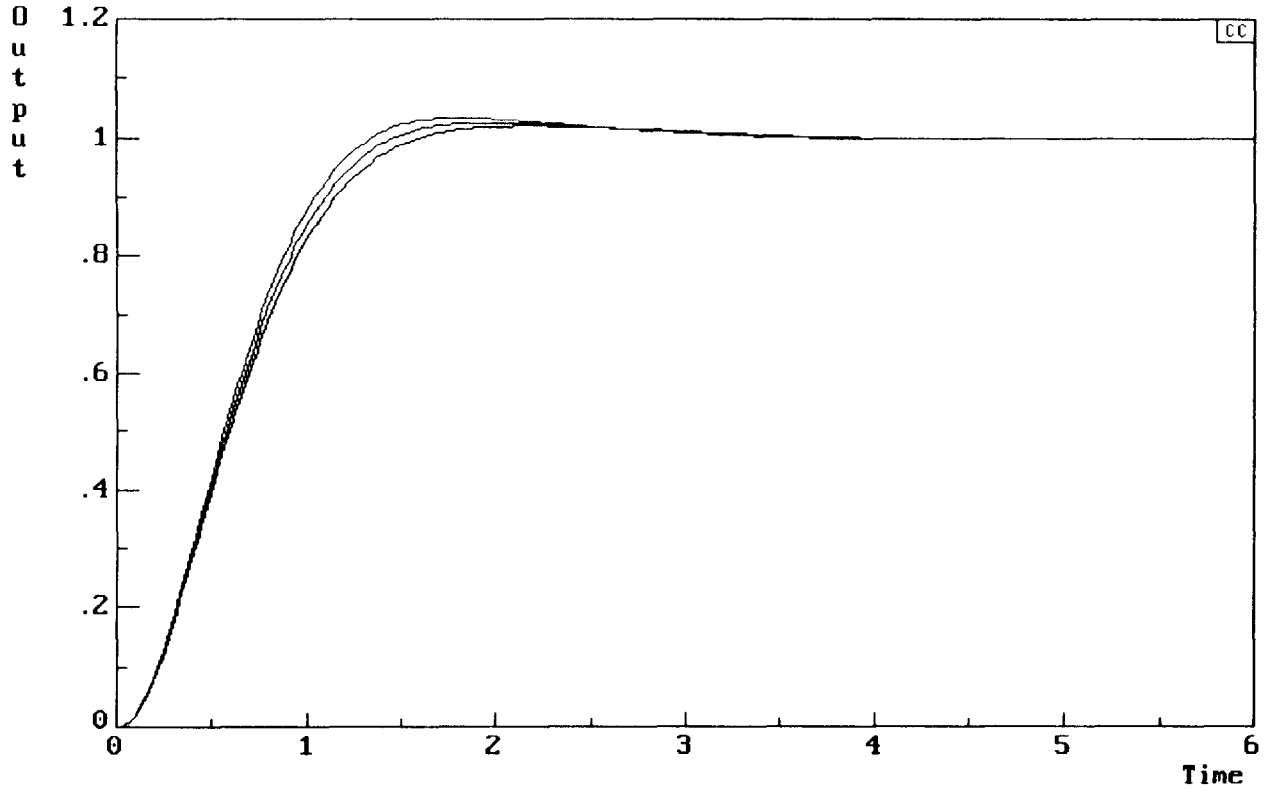


Figure 5.14 Family Of Transient Responses Of The Fixed-Configuration PID Control System For Plant $G_4(S)$ With 0.2 Sec. Delay, And For A Loop-Gain Band Of $\pm$ 5% Relative To Its Value At The Minimum Sensitivity Of The Dominant Closed-Loop Poles.

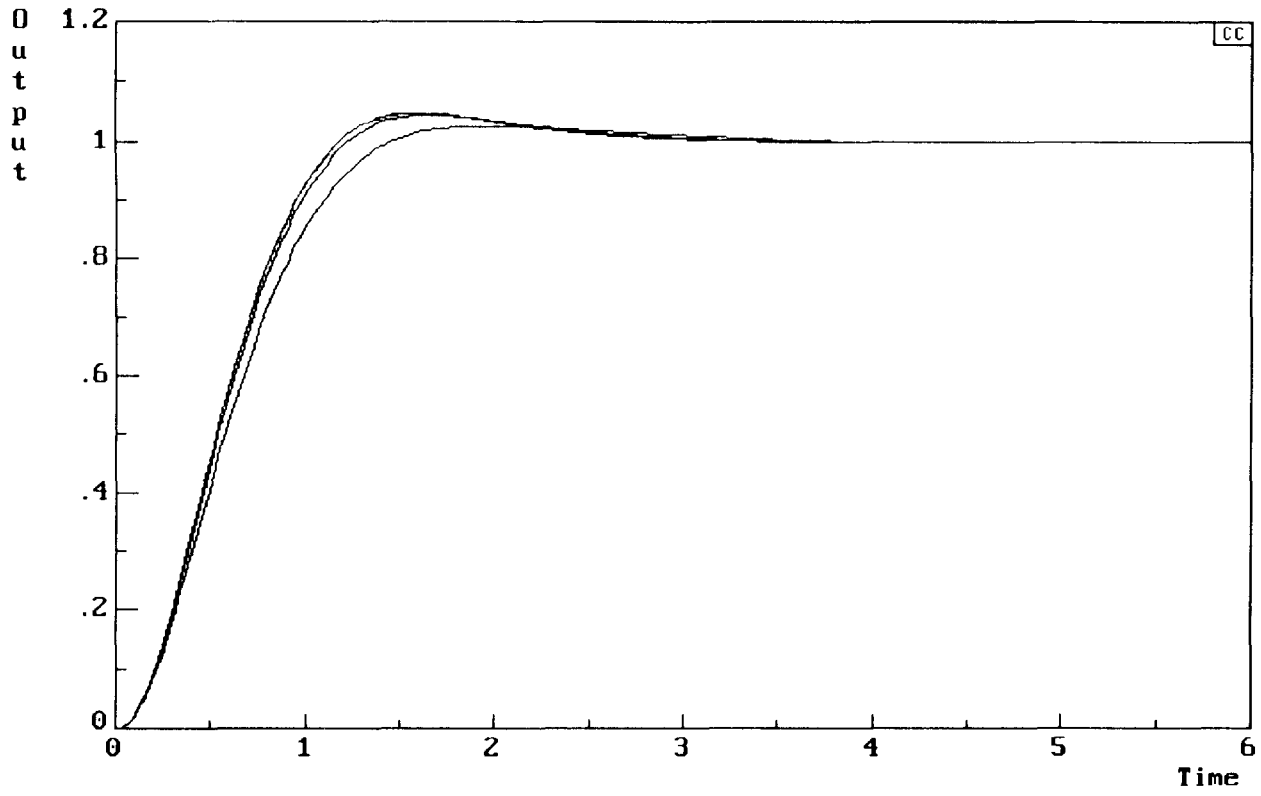


Figure 5.15 Delay-Adjusted Family Of Transient Responses Of The Fixed-Configuration PID Control System For Plant With Delays Of 0.0, 0.05, 0.1 And 0.2 Sec., When The Loop-Gain Is Set At The Minimum Sensitivity Of The Dominant Closed-Loop Poles.

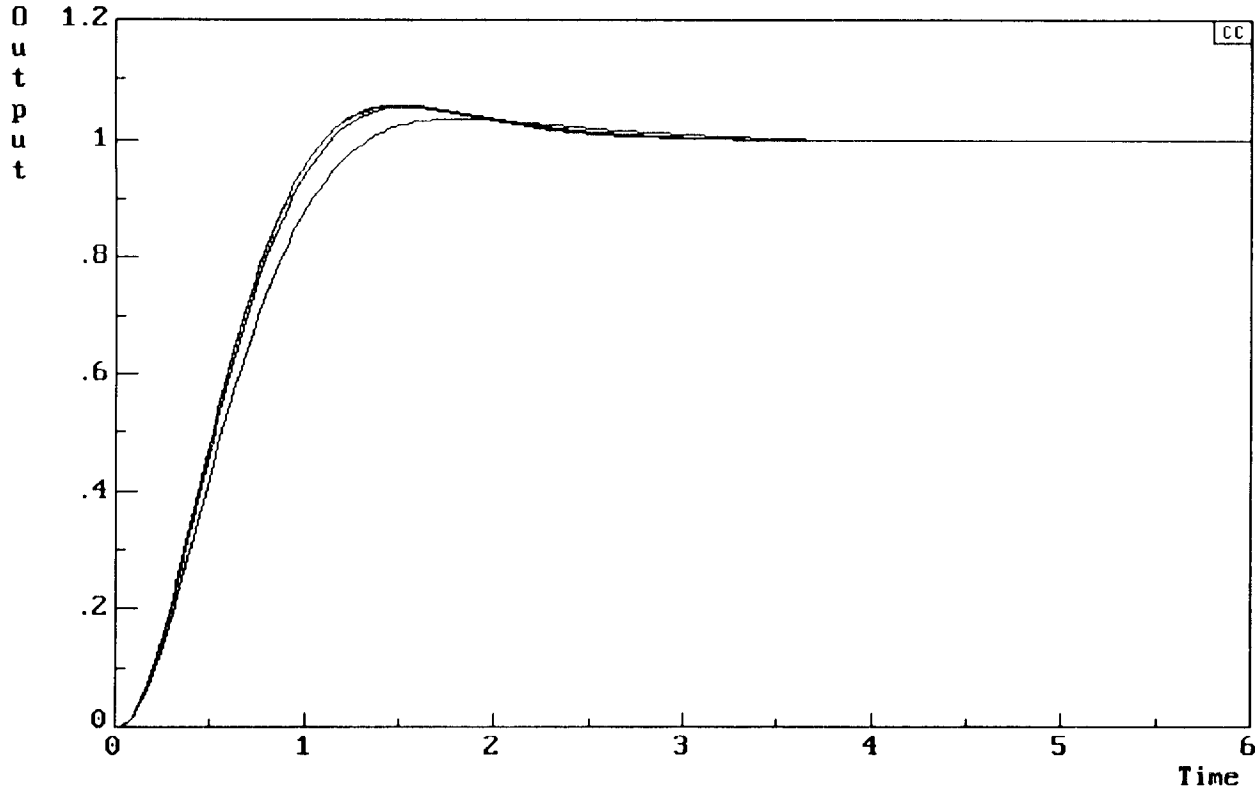


Figure 5.16 Delay-Adjusted Family Of Transient Responses Of The Fixed-Configuration PID Control System For Plant With Delays Of 0.0, 0.05, 0.1 And 0.2 Sec., When The Loop-Gain Is Set At 5% Below Its Value At The Minimum Sensitivity Of The Dominant Closed-Loop Poles.

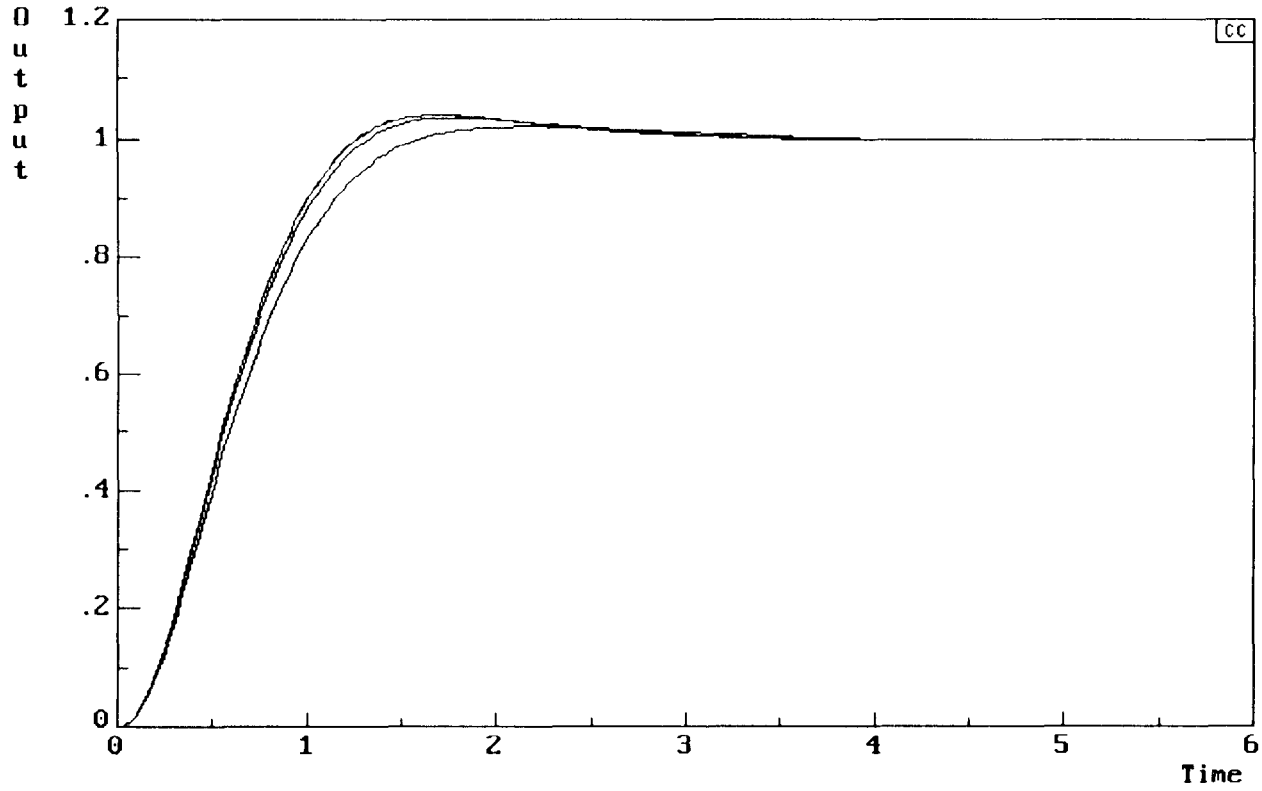


Figure 5.17 Delay-Adjusted Family Of Transient Responses Of The Fixed-Configuration PID Control System For Plant With Delays Of 0.0, 0.05, 0.1 And 0.2 Sec., When The Loop-Gain Is Set At 5% Above Its Value At The Minimum Sensitivity Of The Dominant Closed-Loop Poles.

Chapter 6

Design Of Robust Control Systems For Plants With Uncertainties Using The Developed Sensitivity-Based CAD System

6.1 Introduction

The pervious chapter has established a comprehensive procedure of how the developed sensitivity-based algorithm together with the derived off-line "KBSC" graphic knowledge-base can be used as both efficient and effective CAD system for deterministic plants. Some fundamental aspects of the CAD system have been demonstrated successfully through its application on a case study of a typical autopilot control system. The CAD system has produced an optimum control system design in the form of fixed-configuration controller. The results of extensive testing and evaluations have shown that such controller maintains satisfactory robust qualities of the control system in the event of fairly wide variations of both gain and delay of the plant.

It was shown that, while the gain variations re-locate the closed-loop poles on the system of root-loci (for unity feedback otherwise the closed-loop zeros are also re-located on the same root-loci system for weighted feedback), the delay variations

fundamentally re-shape the root-loci which imply considerable modifications to the root sensitivities.

Variations of gain and delay, however, represent only limited aspects of uncertainties of real plants operating in the real environment.

The first part of this chapter is, therefore, devoted to modelling plants with parameter uncertainties using the developed S-B algorithm. Here the parameter uncertainties are directly translated in terms of uncertainty of the exact positions of the poles and zeros of the plant in the S-plane. In practice, however, the ranges of such uncertainties are limited and can invariably be determined. It follows, therefore, that plant perturbations or uncertainties may be interpreted in terms of poles and zeros located any where within finite fuzzy zones in the S-plane. In general, these fuzzy zones could take any shapes which lay within circular discs of finite radius- as functions of the range and type of uncertainty- centred on fixed points in the general area of the S-plane which represent expected positions. For poles and zeros confined to the real-axis of the S-plane, however, the fuzzy discs collapse into there relevant horizontal diameters as the limiting case; hence maintaining the principle of physical realisability.

It is obvious from the above interpretations that robust control system design procedure for plants with uncertainties present a considerably more complex problem compered with that of deterministic plants. The objective of the rest of this chapter is, therefore, targeted to the design strategy adopted and to the role of the S-B algorithm in providing such strategy and the method of evaluation leading to either a fixed-configuration controller or an adaptive controller design. The degree of automation in selecting coarse and fine windows of computations within prescribed sensitivities together with the flexible adaptability embodied in the algorithm, in addition to the other properties and facilities already discussed in the pervious chapters, are revealed to be crucial for the algorithm's suitability for such applications.

6.2 Modelling Of Real-life Plants' uncertainties

In most practical applications such as in industry, plant uncertainties are invariably present. there are three major causes to their presence:

1. Modelling errors, arising from inaccurate representation at high frequencies, from lack of all the information necessary for parametric calculations or from difficulties in their exact measurements.
2. Unpredictable variations in environmental conditions.
3. Changes in components or systems' characteristics due to either age degradation or due to relatively wide-range of operating conditions.

Robust design of control systems to such plants, would therefore requires the incorporation of an uncertainty criterion from the outset in order to minimise performance degradation and to virtually eliminate the probability of instability.

To illustrate the process of modelling plant uncertainties, the aircraft pitch- plane dynamics including its actuator referred to in the case study, section 5.2.1, is considered here as an example of a real- life plant, where the expected positions of the pole/zero configuration in the S-plane is given by the deterministic model equation (5.1), namely:

* A real pole at S = -4.0
* A pair of complex conjugate poles at S = $-1.0 \pm j1.73$

As mentioned in the introduction, parametric perturbations cause unpredictable migrations of the plant roots within fuzzy discs whose boundary dimensions are related to both the range and the nature of the uncertainties. It follows, therefore, that for the same degree and type of parameter uncertainties, the size of the fuzzy disc associated with a particular root of the plant is a function of its position in the S-plane; i.e., a

function of its distance from the origin. Such function is essentially mirrored for any pair of complex-conjugate roots (for all physically realisable plants) but is not necessarily the same function for other roots of the same plant. The same principal applies to the plant roots lying on the real-axis except that the shape of the fuzzy disc becomes infinitesimally narrow surrounding its horizontal diameter. Figure 6.1 shows the modelling of uncertainties in the aircraft pitch-plane dynamics including its actuator in terms of variations of its roots within fuzzy discs.

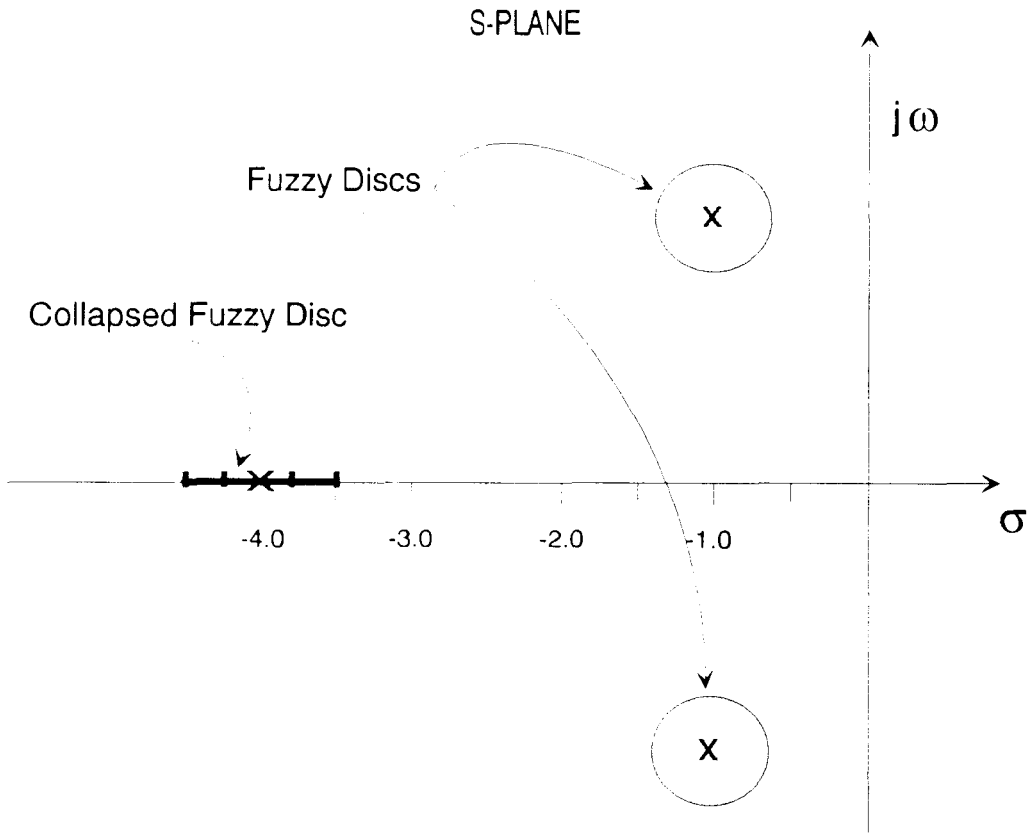


Figure 6.1 Variation Of The Pole/Zero Configuration Of An Aircraft Pitch-Plane Dynamics And Its Actuator Within Fuzzy Discs (Shown Magnified) To Model Uncertainties.

Effectively, for every specific variation of the root's positions, there will be an associated system of root-loci and root-sensitivity diagrams which encompass plant performance on closed-loop as functions of loop gain and/or delays. Therefore, to

carry out design investigations, families of these two respective loci are needed. Although the number of each family members extends to infinity, they are infinitesimally close to each other to form **thick root-contour and root-sensitivity diagrams**. It is, therefore, sufficient to consider a finite number of these members which form the boundary surface of these thick contour diagrams without excessive loss of information nor accuracy. These can be obtained by selecting plant roots' positions on the perimeters of their respective fuzzy discs, spaced at regular intervals and with an arbitrary order as illustrated in figure 6.2. Their density on the respective perimeters can be increased or reduced in proportion to the size of the largest fuzzy disc of the configuration so as to maintain an acceptable compromise between accuracy of information and speed of computations.

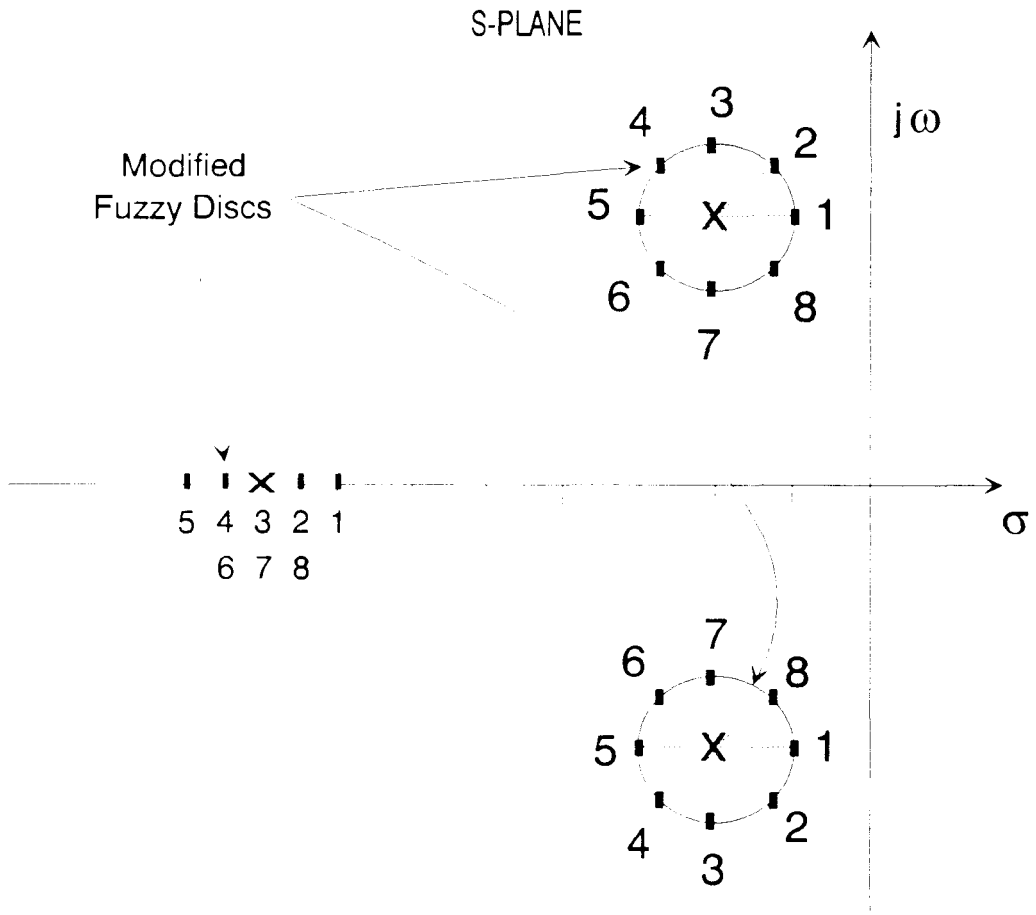


Figure 6.2 Migration Strategy Of The Plant Roots Along The Perimeters Of Their Respective Fuzzy Discs (Shown Magnified).

For the case study of design of a robust control system for the aircraft autopilot considered in this chapter, the plant uncertainty is modelled as follows:

Preliminary Investigations- Case Study "A":

The plant dynamics including the actuator, which have been previously approximated by the deterministic model of equation (5.1), is now assumed to be subjected the following parametric variations when operating in a more realistic environment:

1. The damping ratio of the plant "ξ_1", varies in the range of 0.48 to 0.52,

2. The undamped natural frequency of the plant "ω_1" varies in the range of 1.91 to

2.08 rad./s,

3. The time constant of the actuator "τ_1"is nearly constant at o.2 sec.,

4. The plant gain "K_p" varies in the range of $\pm$5% around its nominal value of 1.0,

5. The plant delay "T_d" varies in the range of 0.0 to 0.05 sec.

The variations of "ξ_1" and "ω_1" translate into variations of the plant's complex poles in the S-plane. The twin-mirror imaged-regions of their variations can be determined as the enclosed zones (referred to in this thesis as the actual fuzzy zones) between two sets of loci: The first set represent the upper and lower boundaries of the constant ξ_1-loci as defined by "ξ_1" range, and the second set represent the upper and lower boundaries of the constant ω_1-loci as defined by "ω_1" range. Figure 6.3 shows the intersections of these two sets of loci which define the boundaries of the actual fuzzy zones. Circular discs- centred on the expected positions of the complex poles (at -1.0 $\pm$ j1.73)- can then be found which enclose the actual fuzzy zones completely with a minimum radius. These discs can then be used to replace the irregularly shaped actual

fuzzy zones as an approximation. In this way, a consistent migration strategy of the plant roots along their respective disc perimeters (see figure 6.2) may be adopted and included in the algorithm for modelling uncertainties.

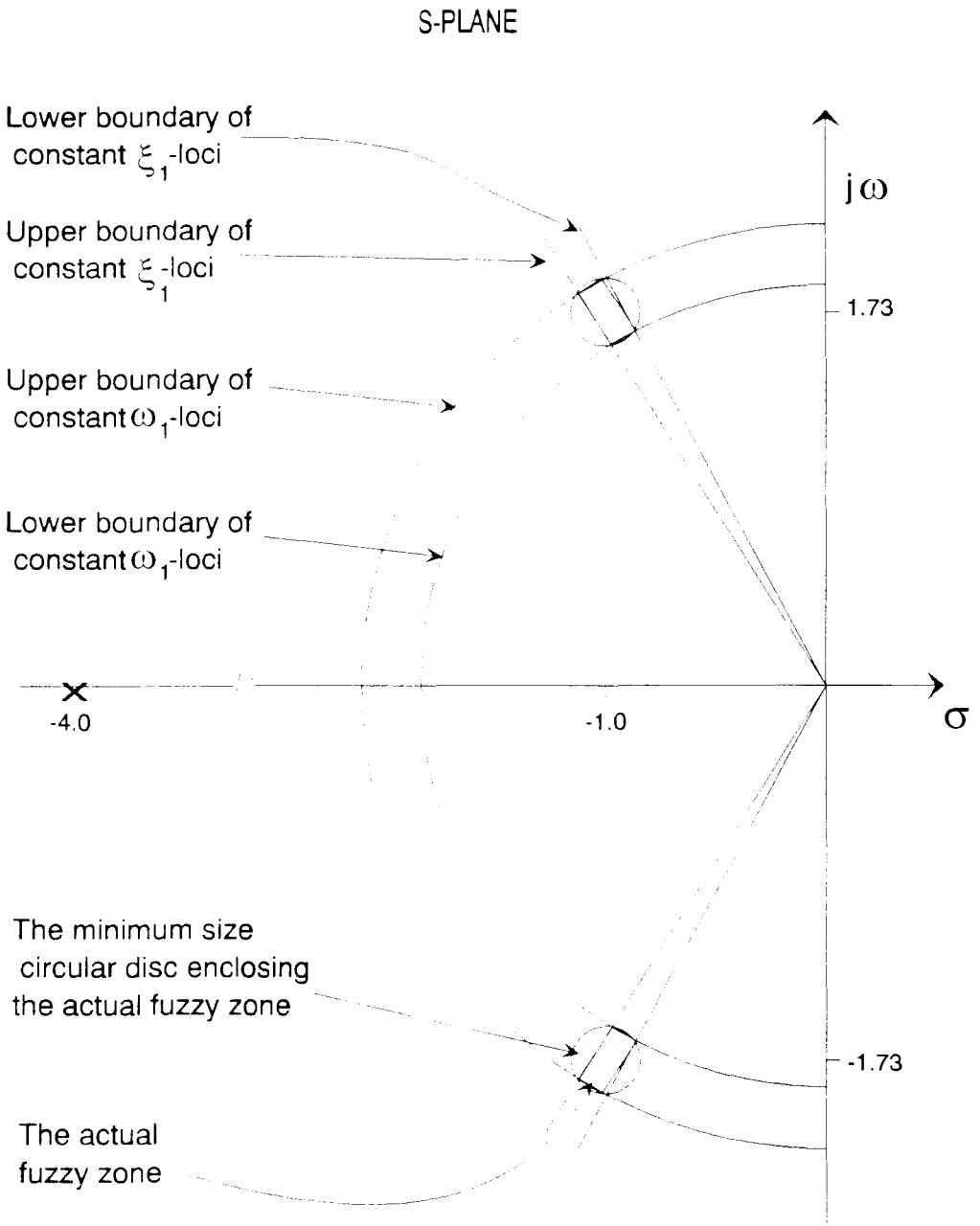


Figure 6.3 Determination Of The Actual Fuzzy Zones Of The Plant's Complex Poles Variations And Their Approximation By Fuzzy Circular Discs.

For this case study, the radius of the fuzzy disc within which the plant's complex poles vary was found to be approximately equal to 0.1. Four migration positions of the poles along their respective disc perimeter were also found to be sufficient to construct the boundary surface of the contour diagrams.

Figures 6.4 and 6.6 show the boundary surface of the root-contour diagrams superimposed onto the performance specifications loci-of figure 5.2- as a background, while figures 6.5 and 6.7 show the boundary surface of the corresponding root-sensitivities as functions of the loop gain for plant delays of 0.0 and 0.05 seconds respectively.

These surface boundary diagrams represent snapshots of only one method of the algorithm's constructions of the same selected for convenient off-line interpretations. For on-line or fast screen man-machine interactions, however, the algorithm may use a different set of equations as derived in chapter 2 to construct successive cross-sectional discs of the surface boundaries as the loop gain is automatically incremented within a relatively small range for fine inspection or on-line operation. Alternatively, successive cross-sectional composite discs of the surface boundaries can be produced by the algorithm as both the loop gain and plant delay are automatically incremented within fine ranges.

Preliminary Investigations- Case Study "B":

Using the same plant of case study "A", except that it is now assumed that the plant is subjected to even larger parametric variations as summarised below:

1. The damping ratio of the plant "ξ_1", varies in the range of 0.46 to 0.55,
2. The undamped natural frequency of the plant "ω_1" varies in the range of 1.82 to 2.16 rad./s,

While "τ_1" and the variations of "K_p" and "T_d" are kept the same as those in case study "A".

Modelling of plant uncertainties follow a similar scheme as described in connection with case study "A". The radius of the fuzzy disc within which the plant's complex poles vary was found to be approximately equal to 0.2. Four migration positions of the poles along their respective disc perimeter were also found to be sufficient to construct the boundary surface of the contour diagrams.

The resulting snapshots of the screen display produced by the algorithm are shown in figures 6.8 and 6.10 for the boundary surface of the root contour diagrams superimposed onto the performance specifications loci-of figure 5.2- as a background, while figures 6.9 and 6.11 show the boundary surface of the corresponding root-sensitivities as functions of the loop gain for plants delays of 0.0 and 0.05 seconds respectively.

On closer examination of the composite screen displays (or the reproduced graphics) for both case studies, it is evident that the proportional-term controller cannot adequately satisfy all the specifications stipulated in section 5.2.2., least of all the steady-state error which has been already anticipated earlier.

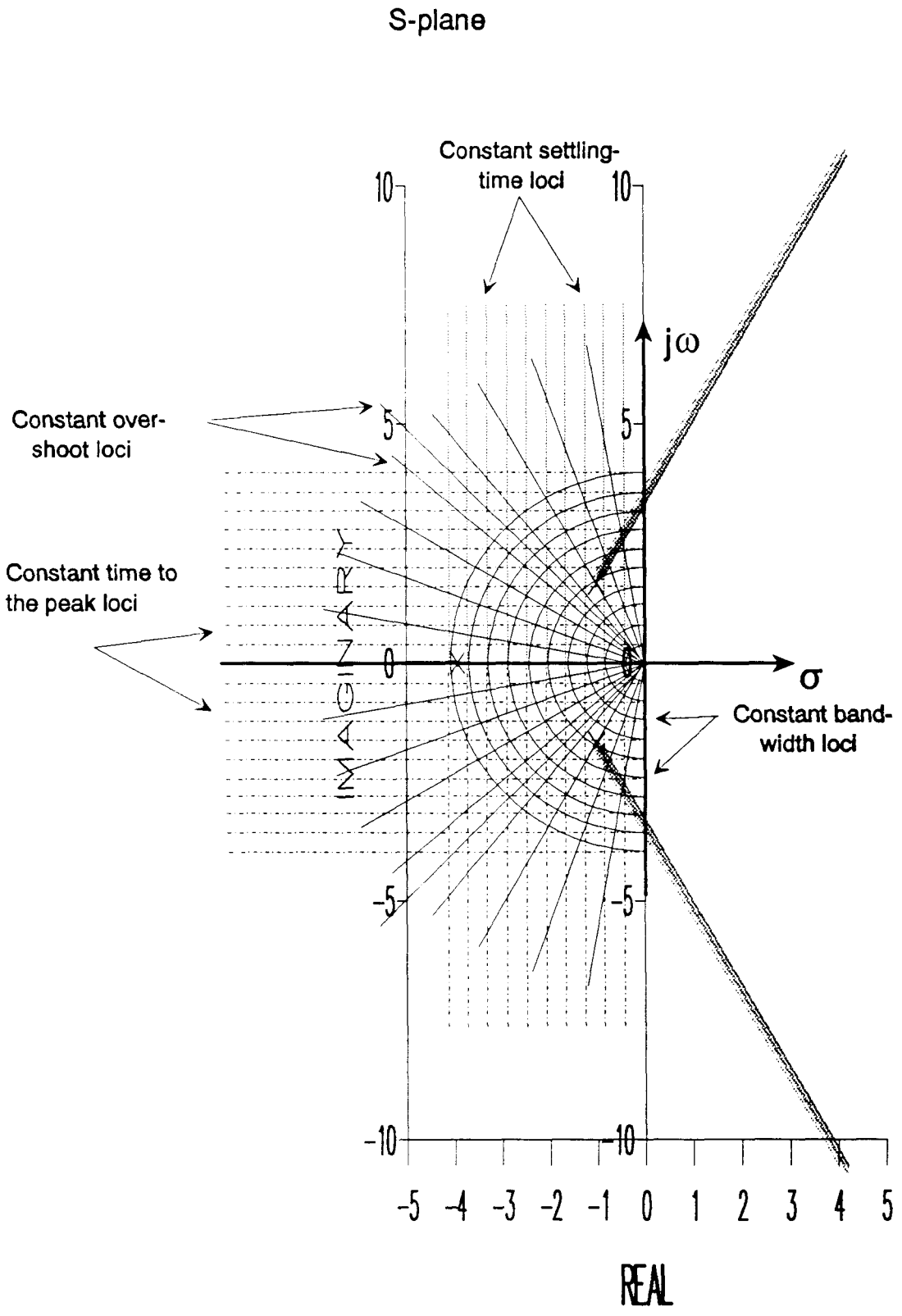


Figure 6.4 Boundary Surface Of The Root-Contour Diagrams Of The Proportional-Term Control System For Plant Of Case-Study "A" With Variations Of The Parameters "ξ_1", "ω_1", "K_p" While The Delay Is Kept Constant At 0.0 sec. (Performance Specifications Loci Are Superimposed)

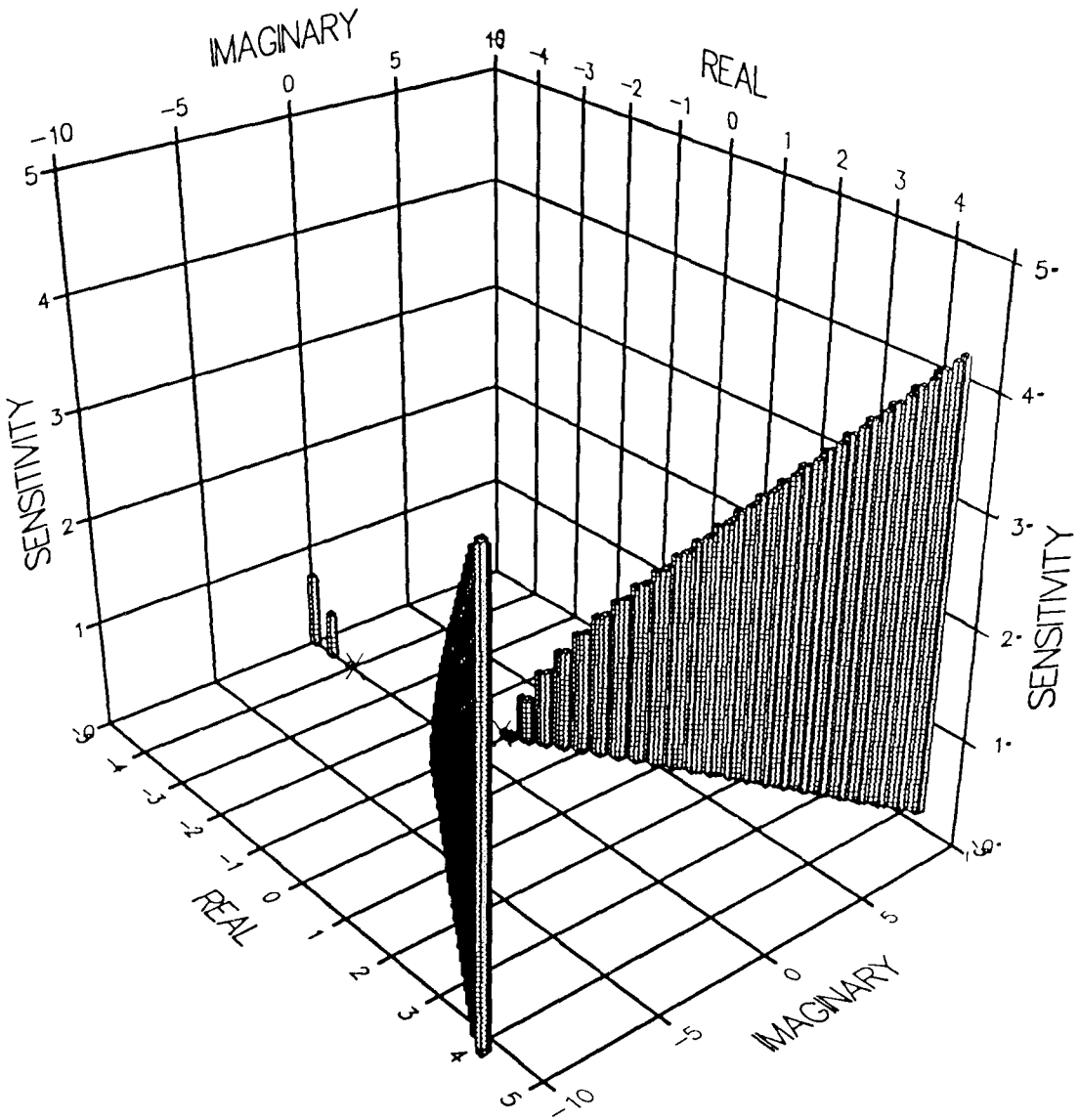


Figure 6.5 Boundary Surface Of The Root-Sensitivity Profiles Of The Proportional-Term Control System For Plant Of Case-Study "A" With Variations Of The Parameters "ξ_1", "ω_1", "K_p" While The Delay Is Kept Constant At 0.0 sec.

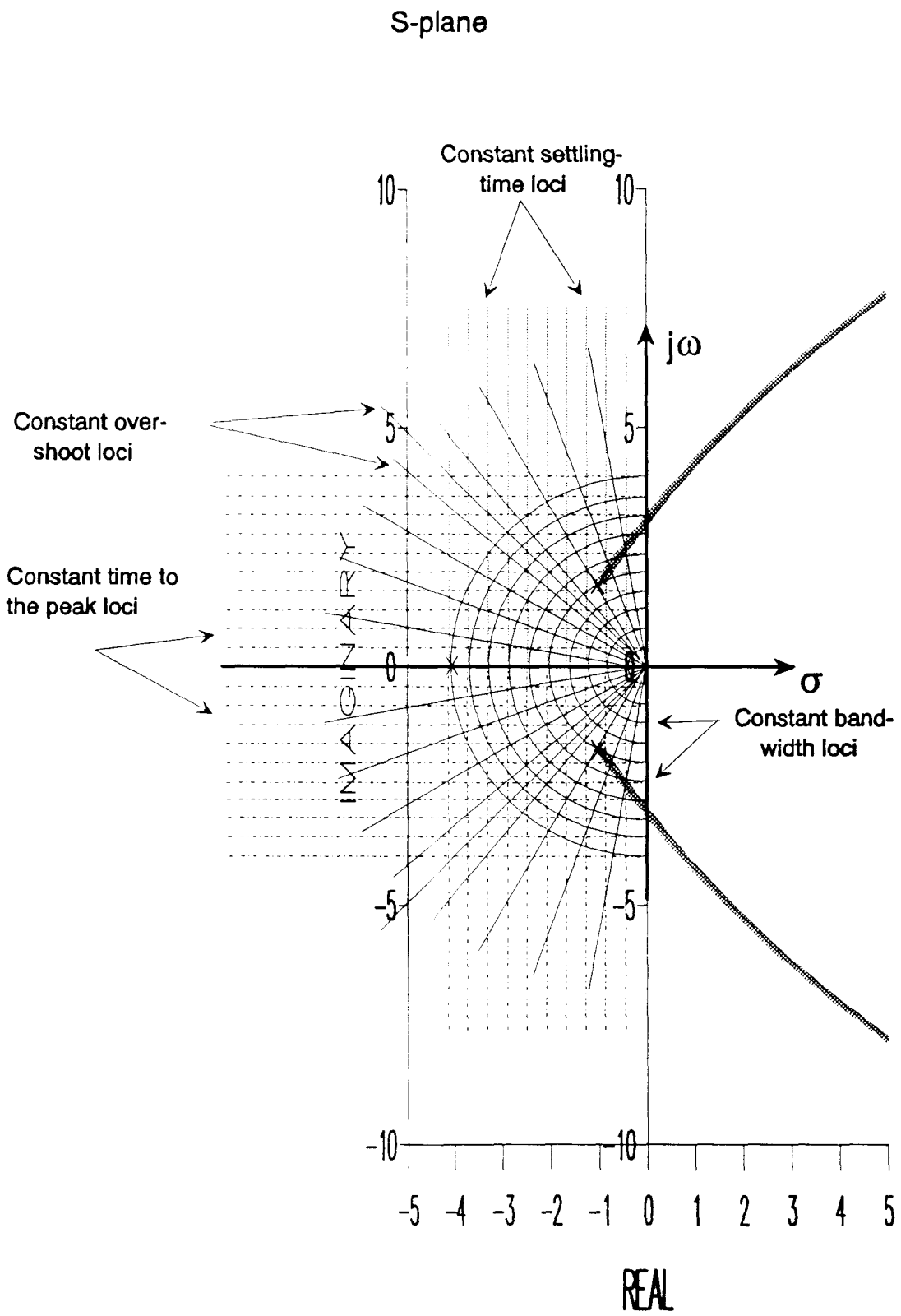


Figure 6.6 Boundary Surface Of The Root-Contour Diagrams Of The Proportional-Term Control System For Plant Of Case-Study "A" With Variations Of The Parameters "ξ_1", "ω_1", "K_p" While The Delay Is Kept Constant At 0.05 sec. (Performance Specifications Loci Are Superimposed)

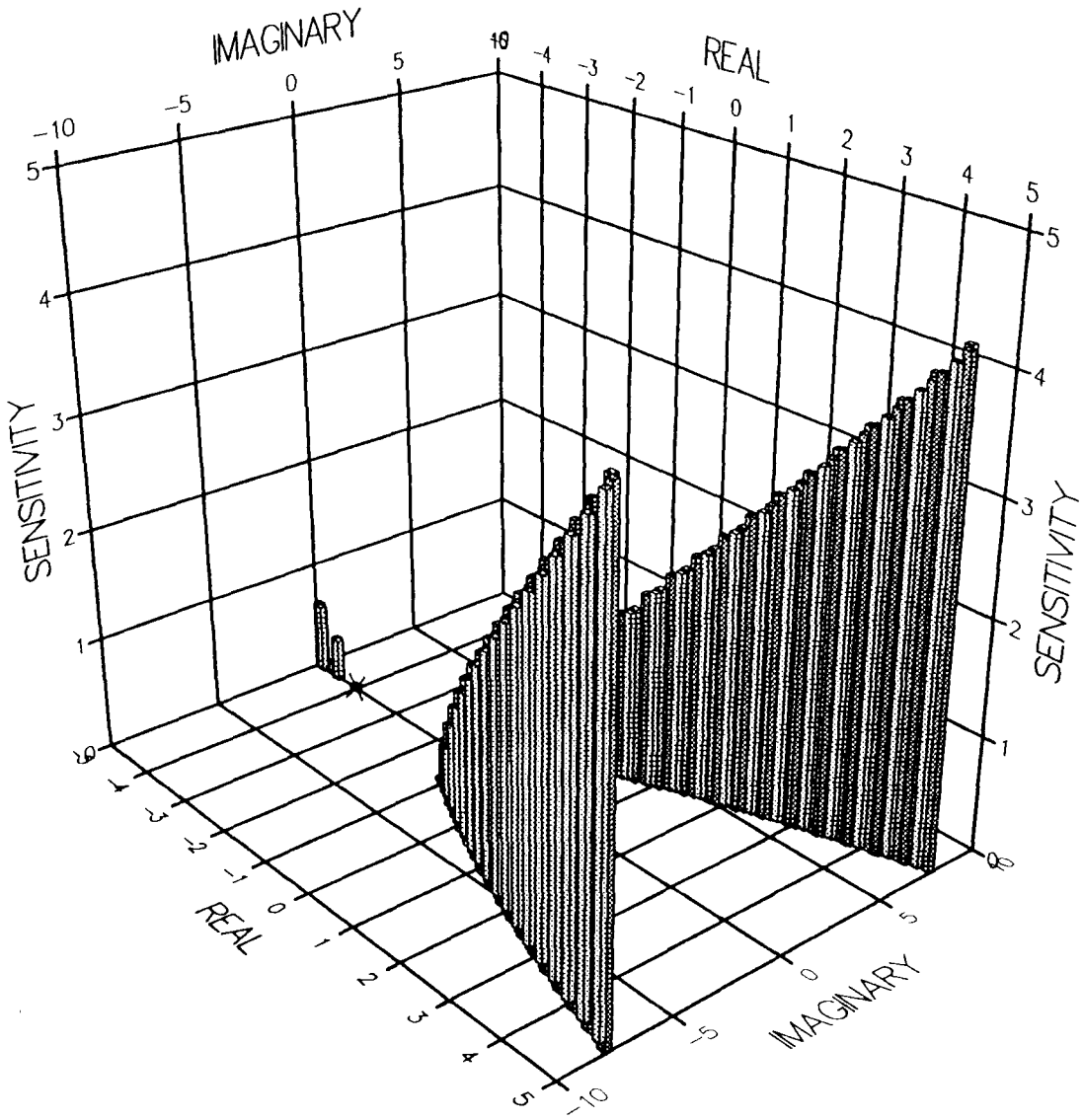


Figure 6.7 Boundary Surface Of The Root-Sensitivity Profiles Of The Proportional-Term Control System For Plant Of Case-Study "A" With Variations Of The Parameters "ξ_1", "ω_1", "K_p" While The Delay Is Kept Constant At 0.05 sec.

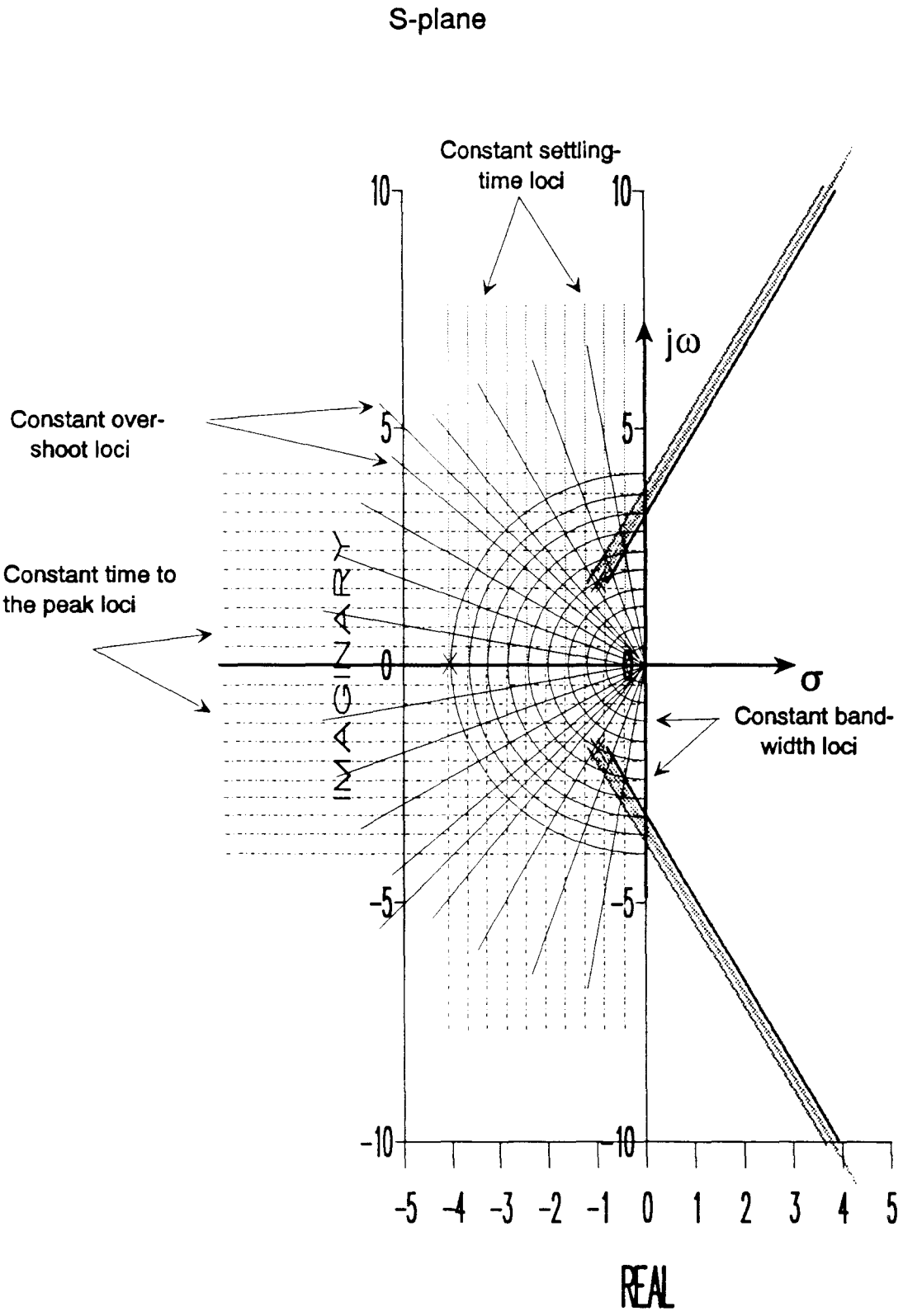


Figure 6.8 Boundary Surface Of The Root-Contour Diagrams Of The Proportional-Term Control System For Plant Of Case-Study "B" With Variations Of The Parameters "ξ_1", "ω_1", "K_p" While The Delay Is Kept Constant At 0.0 sec. (Performance Specifications Loci Are Superimposed)

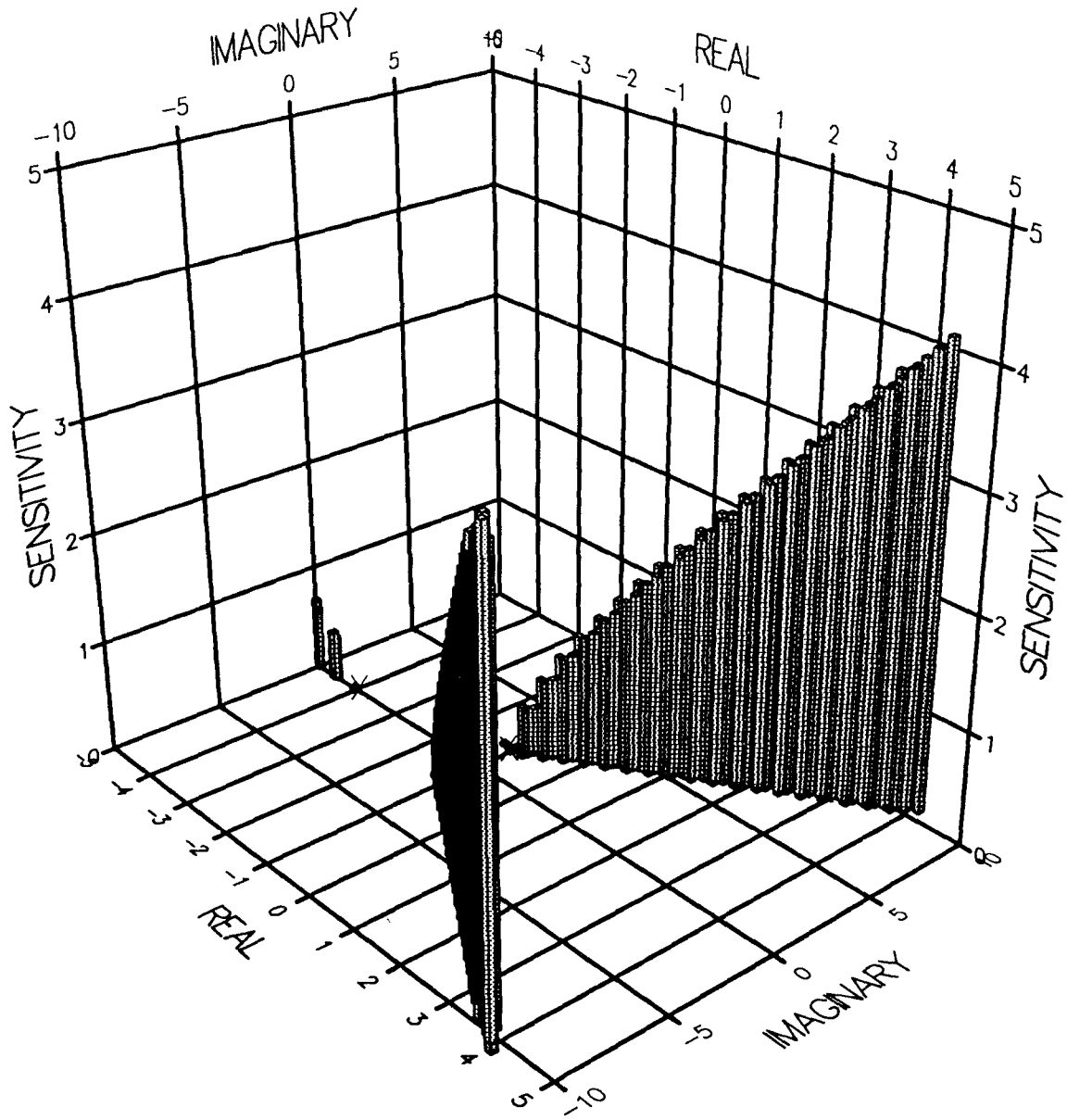


Figure 6.9 Boundary Surface Of The Root-Sensitivity Profiles Of The Proportional-Term Control System For Plant Of Case-Study "B" With Variations Of The Parameters "ξ_1", "ω_1", "K_p" While The Delay Is Kept Constant At 0.0 sec.

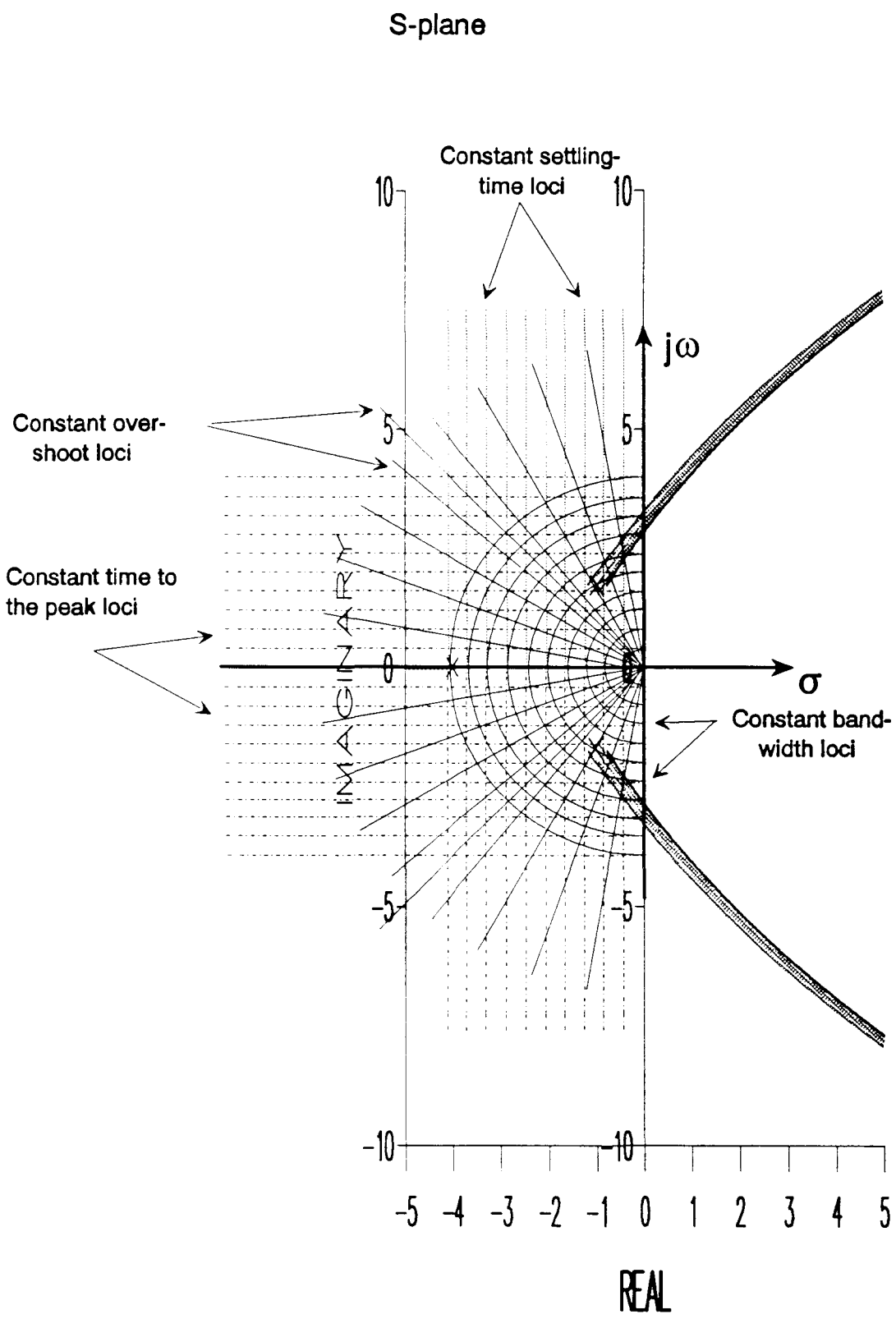


Figure 6.10 Boundary Surface Of The Root-Contour Diagrams Of The Proportional-Term Control System For Plant Of Case-Study "B" With Variations Of The Parameters "ξ_1", "ω_1", "K_p" While The Delay Is Kept Constant At 0.05 sec. (Performance Specifications Loci Are Superimposed)

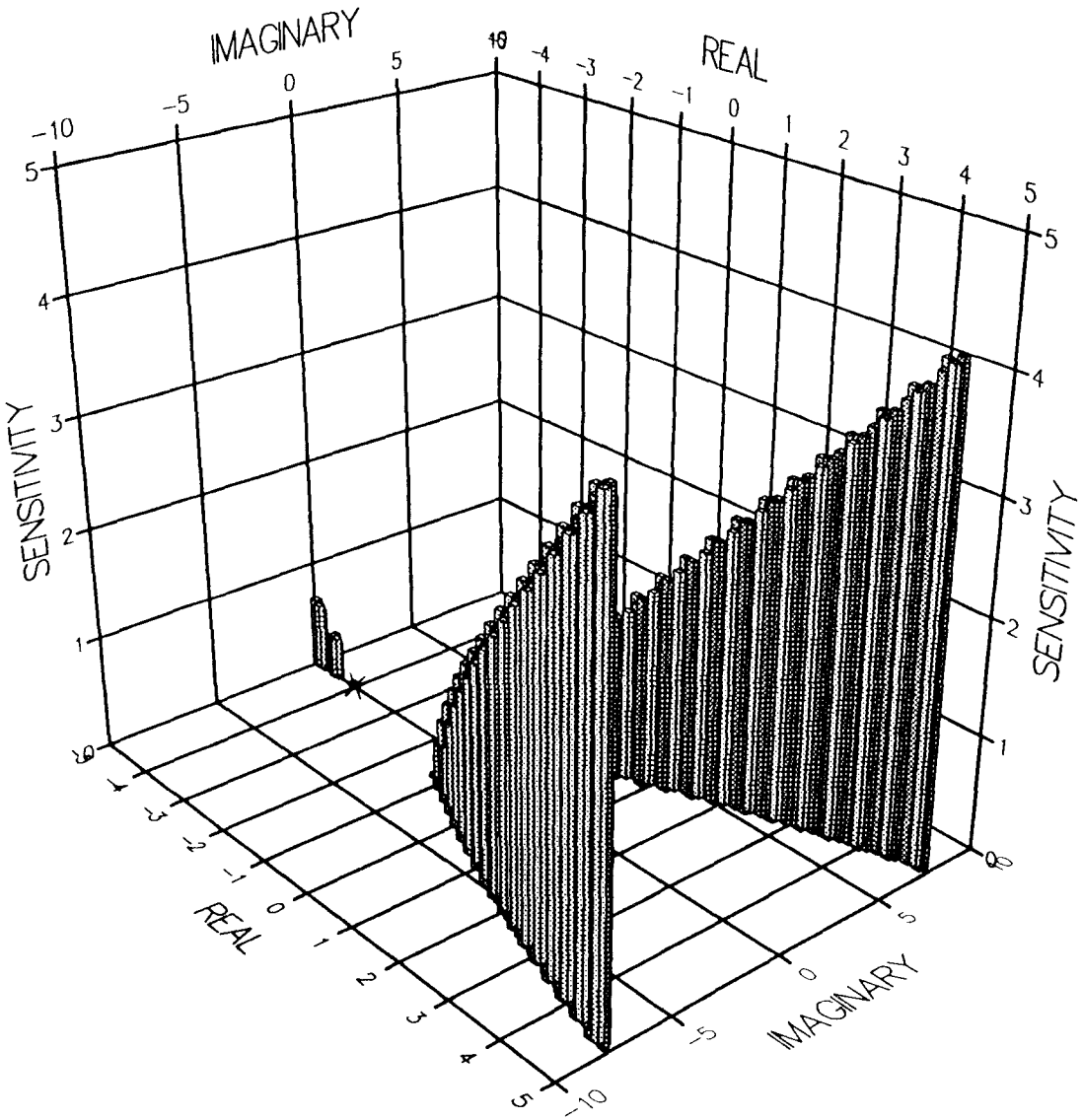


Figure 6.11 Boundary Surface Of The Root-Sensitivity Profiles Of The Proportional-Term Control System For Plant Of Case-Study "B" With Variations Of The Parameters "ξ_1", "ω_1", "K_p" While The Delay Is Kept Constant At 0.05 sec.

6.3 CAD Procedure For A Fixed-Configuration Optimum Controller Design

Focusing on the plant of case study "A" of section 6.2 and adopting a similar strategy as outlined in section 5.4, has lead to a fixed-configuration optimum controller design having the transfer function "$G_{CF}(S)$" as described by equation 5.12. Snapshots of the resulting boundary surfaces of the root contour diagrams of the optimum autopilot design are shown in figures 6.12 and 6.14- superimposed onto the performance specifications loci as a background- for plant delays of 0.0 and 0.05 seconds respectively. Whereas figures 6.13 and 6.15 show snapshots of the boundary surfaces of the corresponding root sensitivities profiles.

It can be seen from these sets of boundary surface graphs that the fixed-configuration controller has brought the system characteristics very close to its target specifications (outlined in section 5.2.2). It did not, however, improve the system robustness as manifested by the thickness of the boundary surfaces. Indeed, the root contours show bulges almost at the optimum positions of the dominant closed-loop poles which predict relatively wide variations of the system time-domain responses as the plant parameters vary.

A more detailed scrutiny of the system behaviour in the vicinity of its optimum operating region can be conducted using the quantitative results (obtained from the CAD system) summarised in tables 6.1A and 6.1B for plant delays of 0.0 and 0.05 seconds respectively. The results in each table are divided into three sets which are classified according to the loop-gain "K" adjustments:

i) "K" set at the system's minimum sensitivity of the dominant closed-loop poles,

ii) "K" set at approximately 5% below that of (i),

iii) "K" set at approximately 5% above that of (i)

Table 6.1A Results Of The Fixed-Configuration Optimum Control System's Design For A Plant With Parametric Variations, Gain Variations Of ± 5%, And With A Delay Of 0.0 Sec.(Figures Rounded To Four Decimal Places)

Loop Gain Settings*	Plant's Complex Poles Migrations Around The Perimeter Of Their Respective Fuzzy Discs	Sensitivity Of The Dominant Closed-Loop Poles	Bandwidth (rad. s^{-1})	Loop Gain ($K_P K_{CF}$) (rad. s^{-1})	Attenuation Required Outside The Loop For Unity d.c. Gain	Velocity Error Constant K_V (rad. s^{-1})	Steady-State Error For A Unit-Ramp Input	Time-Domain Performance For A Unit-Step Target Input		
								Percent Overshoot	Settling Time (sec.)	Rise Time (sec.)
Lower Threshold	Centre	1.8939	2.8702	306.7360	1.0000	1.8669	0.5356	3.9890	2..3635	0.6341
	East	**2.0087**	2.4238	250.0182	0.9981	1.6006	**0.6247**	3.2450	2.4074	0.8979
	North	**2.8236**	2.0226	202.3548	0.9937	1.1328	**0.8827**	0.5320	**2.6148**	**1.5260**
	West	1.7195	3.0213	296.4272	0.9978	1.7171	0.5824	3.2450	2.2300	0.9190
	South	1.7233	2.8710	269.9313	1.0000	1.7971	0.5564	6.5960	2.1948	0.8592
Optimum	Centre	1.8889	2.9474	324.2554	1.0000	1.9736	0.5067	4.6810	2.3636	0.7500
	East	1.9985	2.4788	263.7830	0.9982	1.6868	0.5928	4.0430	2.4235	0.9776
	North	**2.7184**	2.0768	214.1796	0.9940	1.1990	**0.8340**	0.5320	**2.6580**	**1.3908**
	West	1.7188	3.0670	306.9586	0.9988	1.7781	0.5624	3.0850	2.2261	0.8768
	South	1.7211	2.8660	282.4883	1.0000	1.8807	0.5317	**7.2340**	2.1979	0.8310
Upper Threshold	Centre	1.8947	3.0253	342.7698	1.0000	1.9767	0.5059	5.4790	2.3656	0.6972
	East	**2.0077**	2.5358	277.6751	0.9982	1.7777	0.5625	4.6810	2.4417	0.8028
	North	**2.7677**	2.1358	226.8486	0.9958	1.2700	**0.7874**	0.8510	**2.7133**	**1.2588**
	West	1.7224	3.1126	317.8545	0.9958	1.8412	0.5431	3.4040	2.2232	0.8451
	South	1.7263	2.9162	295.6169	1.0000	1.9681	0.5081	**7.7130**	2.2023	0.7887

* Refers To The Upper & The Lower Boundaries Of The Loop-Gain Band (Of ±5%) Centred On Its Value At The Minimum Sensitivity Of The Dominant Closed-Loop Poles

Table 6.1B Results Of The Fixed-Configuration Optimum Control System's Design For A Plant With Parametric Variations, Gain Variations Of ± 5%, And With A Delay Of 0.05 Sec.(Figures Rounded To Four Decimal Places)

Loop Gain Settings*	Plant's Complex Poles Migrations Around The Perimeter Of Their Respective Fuzzy Discs	Sensitivity Of The Dominant Closed-Loop Poles	Bandwidth (rad. s^{-1})	Loop Gain ($K_P K_{CF}$) (rad. s^{-1})	Attenuation Required Outside The Loop For Unity d.c. Gain	Velocity Error Constant K_V (rad. s^{-1})	Steady-State Error For A Unit-Ramp Input	Time-Domain Performance For A Unit-Step Target Input		
								Percent Overshoot	Settling Time (sec.)	Rise Time (sec.)
Lower Threshold	Centre	1.8860	2.8716	339.4047	0.8952	2.0658	0.4841	4.0430	2.3665	0.7923
	East	**2.0051**	2.4531	274.0389	0.9002	1.7544	0.5700	3.2450	2.3482	0.8979
	North	1.9662	3.0975	406.7106	0.8927	2.2769	0.4392	3.0350	**2.5946**	0.6690
	West	1.7994	3.0383	347.3814	0.8241	2.0123	0.4969	3.7230	2.2027	0.9401
	South	1.7393	2.8170	310.5116	0.8736	1.7971	0.5564	**7.3400**	2.1948	0.8451
Optimum	Centre	1.8809	2.9411	358.8757	0.8898	2.1843	0.4578	4.6810	2.3802	0.7394
	East	1.9915	2.5001	288.8372	0.8099	1.8491	0.5408	0.8510	2.3724	**1.0669**
	North	1.9627	3.1502	420.8437	0.8762	2.3560	0.4244	3.0850	**2.5985**	0.6690
	West	1.7929	3.1083	368.3131	0.8342	2.1335	0.4687	3.9890	2.2098	0.8662
	South	1.7367	2.8665	322.7355	0.8713	1.8807	0.5317	**7.7130**	2.1979	0.8028
Upper Threshold	Centre	1.8879	3.0109	379.4510	0.8849	2.3095	0.4330	5.6380	2.3970	0.6972
	East	1.9984	2.5486	304.4611	0.9185	1.9492	0.5130	4.7340	2.4000	0.8134
	North	1.9652	3.2026	435.4706	0.8868	2.4379	0.4102	4.6810	**2.6045**	0.6250
	West	1.8010	3.1777	390.4939	0.8218	2.2620	0.4421	4.3620	2.2210	0.8134
	South	1.7402	2.9162	335.4375	0.8691	1.9681	0.5081	**8.1910**	2.2023	0.7887

* Refers To The Upper & The Lower Boundaries Of The Loop-Gain Band (Of ±5%) Centred On Its Value At The Minimum Sensitivity Of The Dominant Closed-Loop Poles.

to simulate plant gain variations of about $\pm$ 5%.

Each set is sub-divided into five sets, four of which refer to the individual root-loci forming the boundary surface of the root contour, and the fifth refers to the central root-locus (identical to the deterministic case).

Analysing the system's performance reveals that it can only meet all the specifications **within a narrower zone** of the plant's parameter-variation ranges. Otherwise, relatively **small violations** of the specifications appear almost uniformly across the performance indices (with the exception of the bandwidth requirements) as indicated by the highlighted figures in the tables referred to above. Nevertheless, the general observations, summarised in section 5.5, regarding the trend of the performance indices and the attenuation requirement outside the loop as the plant's delay increases, also concur here.

The results have been subjected to further analysis with the objective of obtaining quantitative measures of the control system's robustness. The designed control system has been simulated (using Program CC version 4, reference [57] and a series of d.c. gain normalised transient responses were obtained for a unit-step pitch target angel. Figures 6.16 to 6.18 show three sets of these transient responses which corresponds to plant's delay of 0.0 seconds, and classified according to the loop-gain adjustments of $\pm$ 5% around its optimum value. Each set consists of five responses; four of which correspond to the migrations of the plant's complex poles around the perimeter of their fuzzy discs, and the fifth corresponds to the plant's complex poles located at the centre of their respective fuzzy discs (identical to the deterministic case). Figures 6.19 to 6.21 show three sets of the transient responses which correspond to plant's delay of 0.05 seconds and classified similarly. these time-domain responses show fairly wide variations relative to each other which indicate that the control system suffers from high sensitivity to plant parameter variations, particularly to those responsible for the

plant's complex poles variations. This conclusion concurs with that predicted by the sensitivity-based CAD system.

Table 6.2A and 6.2B give quantitative measures of the system's sensitivity (or robustness) in terms of the mean deviation and the mean variance, relative to the corresponding deterministic cases as references, when the plant's delay is 0,0 and 0.05 seconds respectively. each table is divided into three sets classified according to the loop-gain adjustments similar to that of tables 6.1A and 6.1B. Each set contains four comparisons which correspond to the migrations of the plant's complex poles around the perimeter of their respective fuzzy discs in the order: east, north, west and south. The ranges of the mean deviation and the mean variance, again, support the above conclusions particularly when compared with those obtained for the optimum control system designed for the deterministic plant: the maximum mean deviation and the maximum mean variance show increases of the order of 11 and 83 folds, respectively, for plant with zero delay, and of the order of 8 and 38 folds, respectively, in comparison with their counterparts.

Table 6.2A Quality Assurance Measures Of The Fixed-Configuration PID Control System In Terms Of Transient Response Sensitivity To Plant Parametric Variations And To Plant Gain Variations Of ± 5%, For 0.0 Sec. Delay.

Loop Gain Settings*	Plant's Complex Poles Migrations Around The Perimeter Of Their Respective Fuzzy Discs	Transient Response Sensitivity To Plant Parametric Variations Relative To Its Deterministic Case.	
		Mean Deviation	Mean Variance
Lower Threshold	East	0.016551	0.000808
	North	**0.061503**	**0.010151**
	West	0.014756	0.000520
	South	0.015461	0.000505
Optimum	East	0.016112	0.000832
	North	0.058711	0.009688
	West	0.016846	0.000708
	South	0.016537	0.000599
Upper Threshold	East	0.015823	0.000858
	North	0.055912	0.009109
	West	0.018778	0.000916
	South	0.017632	0.000703

* Refers To The Upper & The Lower Boundaries Of The Loop-Gain Band (Of ±5%) Centred On Its Value At The Minimum Sensitivity Of The Dominant Closed-Loop Poles.

Table 6.2B Quality Assurance Measures Of The Fixed-Configuration PID Control System In Terms Of Transient Response Sensitivity To Plant Parametric Variations And To Plant Gain Variations Of ± 5%, For 0.05 Sec. Delay.

Loop Gain Settings*	Plant's Complex Poles Migrations Around The Perimeter Of Their Respective Fuzzy Discs	Transient Response Sensitivity To Plant Parametric Variations Relative To Its Deterministic Case.	
		Mean Deviation	Mean Variance
Lower Threshold	East	0.016862	0.000816
	North	0.018759	0.001028
	West	0.017933	0.000749
	South	0.015544	0.000464
Optimum	East	**0.046741**	**0.004731**
	North	0.016213	0.000683
	West	0.016980	0.000662
	South	0.016954	0.000580
Upper Threshold	East	0.015994	0.000794
	North	0.016356	0.000691
	West	0.016053	0.000584
	South	0.018527	0.000728

* Refers To The Upper & The Lower Boundaries Of The Loop-Gain Band (Of ±5%) Centred On Its Value At The Minimum Sensitivity Of The Dominant Closed-Loop Poles.

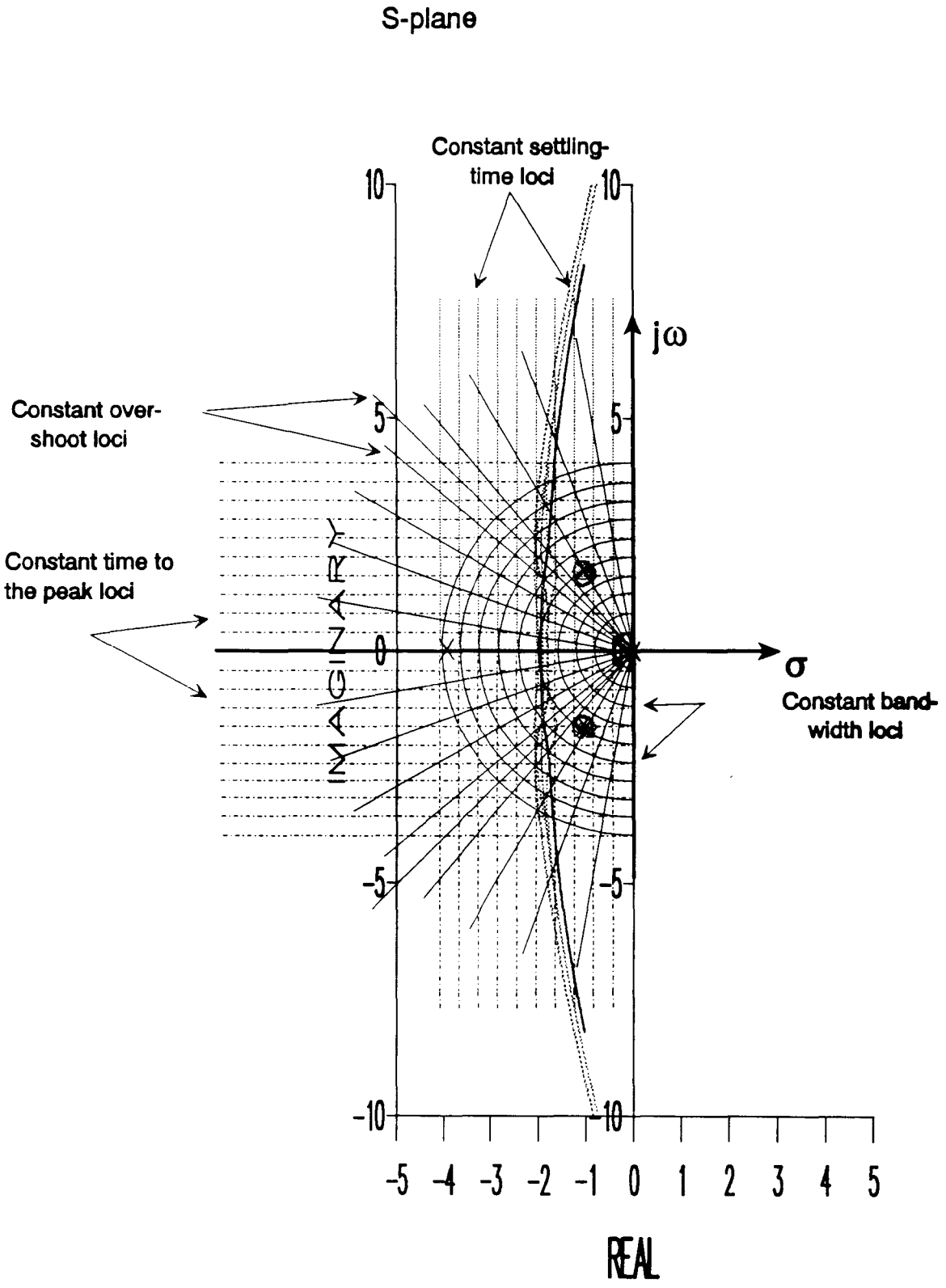


Figure 6.12 Boundary Surface Of The Root-Contour Diagrams Of The Fixed-Configuration Optimum Control System For Plant Of Case-Study "A" With Variations Of The Parameters "ξ_1", "ω_1", "K_p" While The Delay Is Kept Constant At 0.0 sec. (Performance Specifications Loci Are Superimposed)

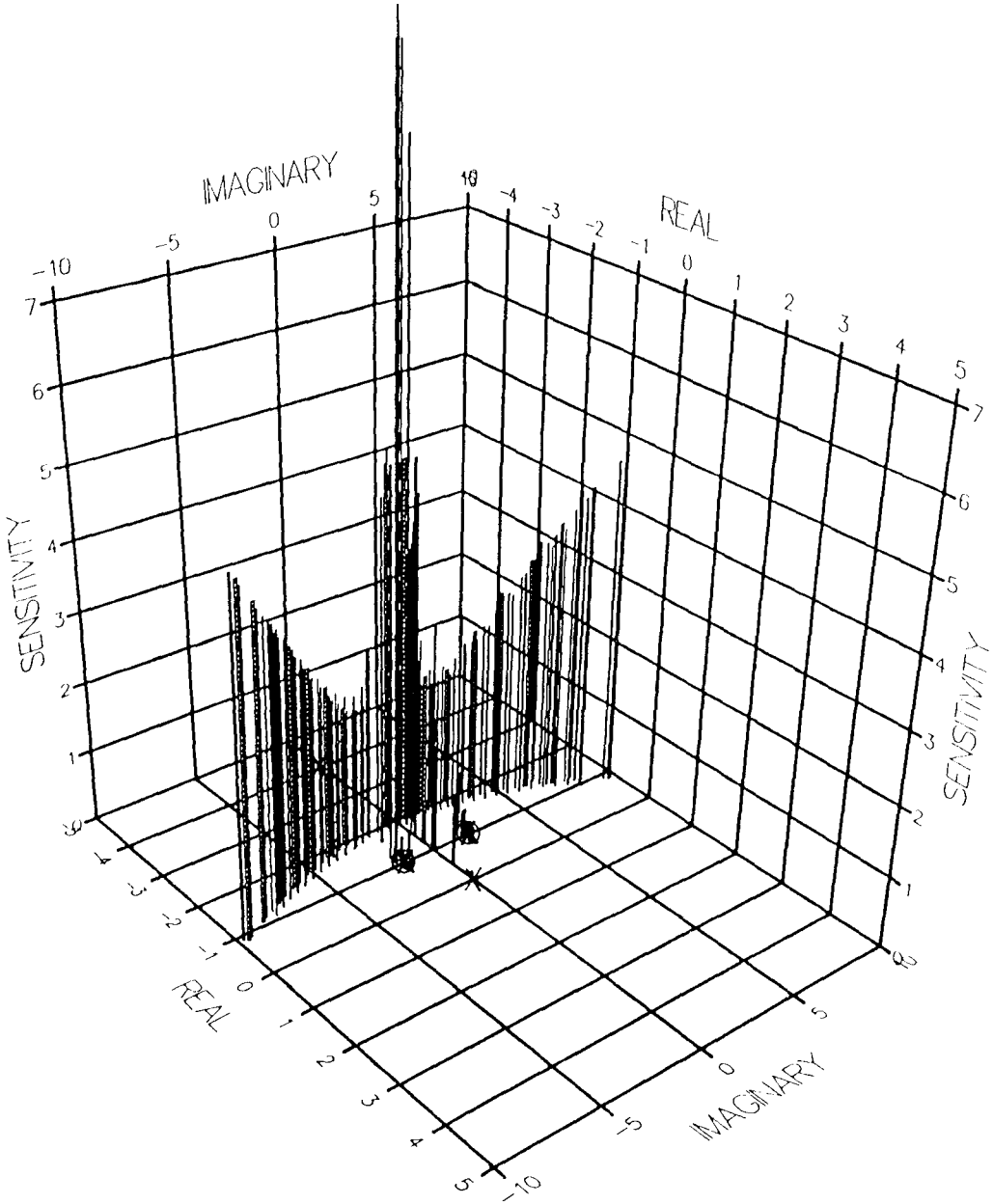


Figure 6.13 Boundary Surface Of The Root-Sensitivity Profiles Of The Fixed-Configuration Optimum Control System For Plant Of Case-Study "A" With Variations Of The Parameters "ξ_1", "ω_1", "K_p" While The Delay Is Kept Constant At 0.0 sec.

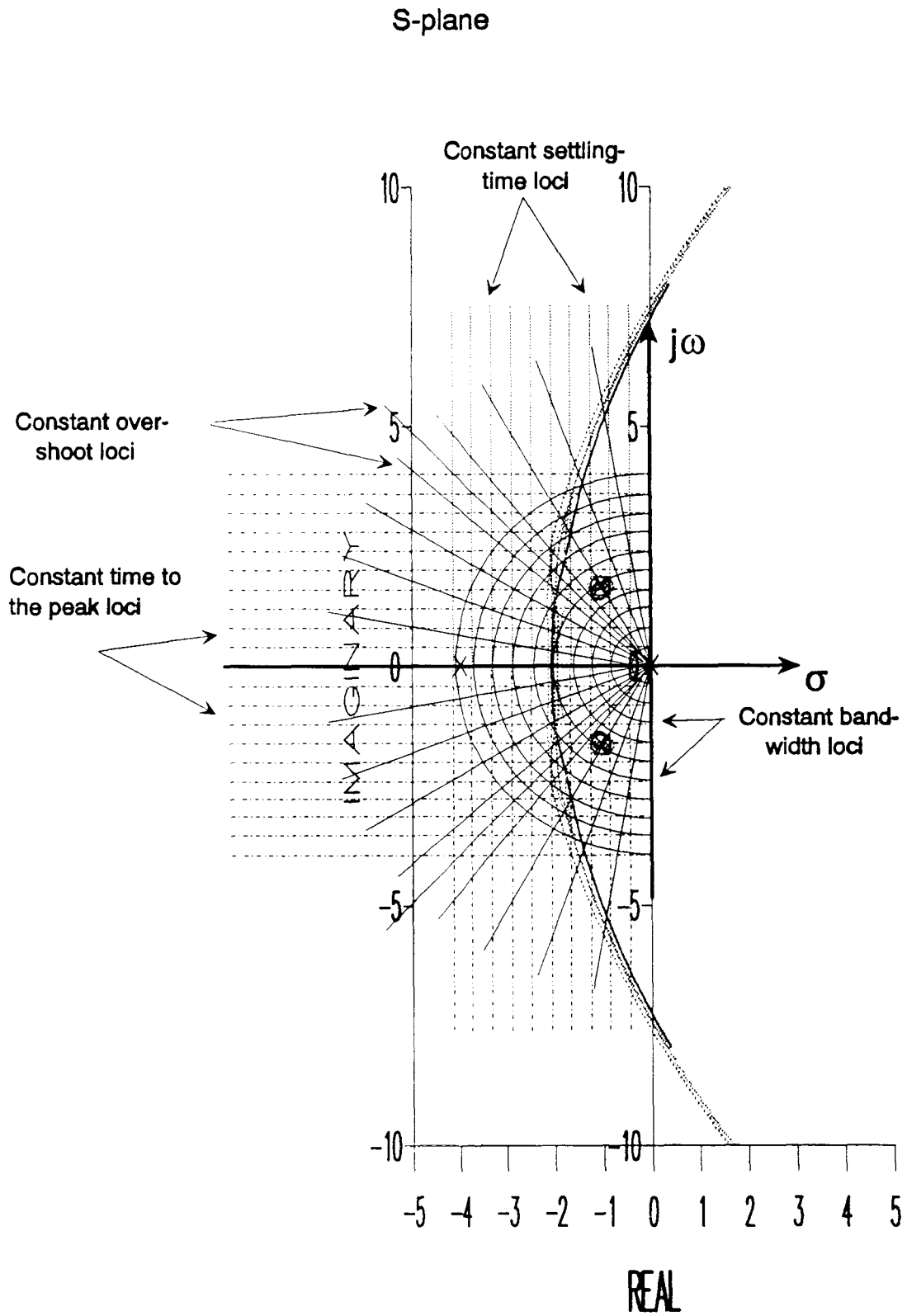


Figure 6.14 Boundary Surface Of The Root-Contour Diagrams Of The Fixed-Configuration Optimum Control System For Plant Of Case-Study "A" With Variations Of The Parameters "ξ_1", "ω_1", "K_p" While The Delay Is Kept Constant At 0.05 sec. (Performance Specifications Loci Are Superimposed)

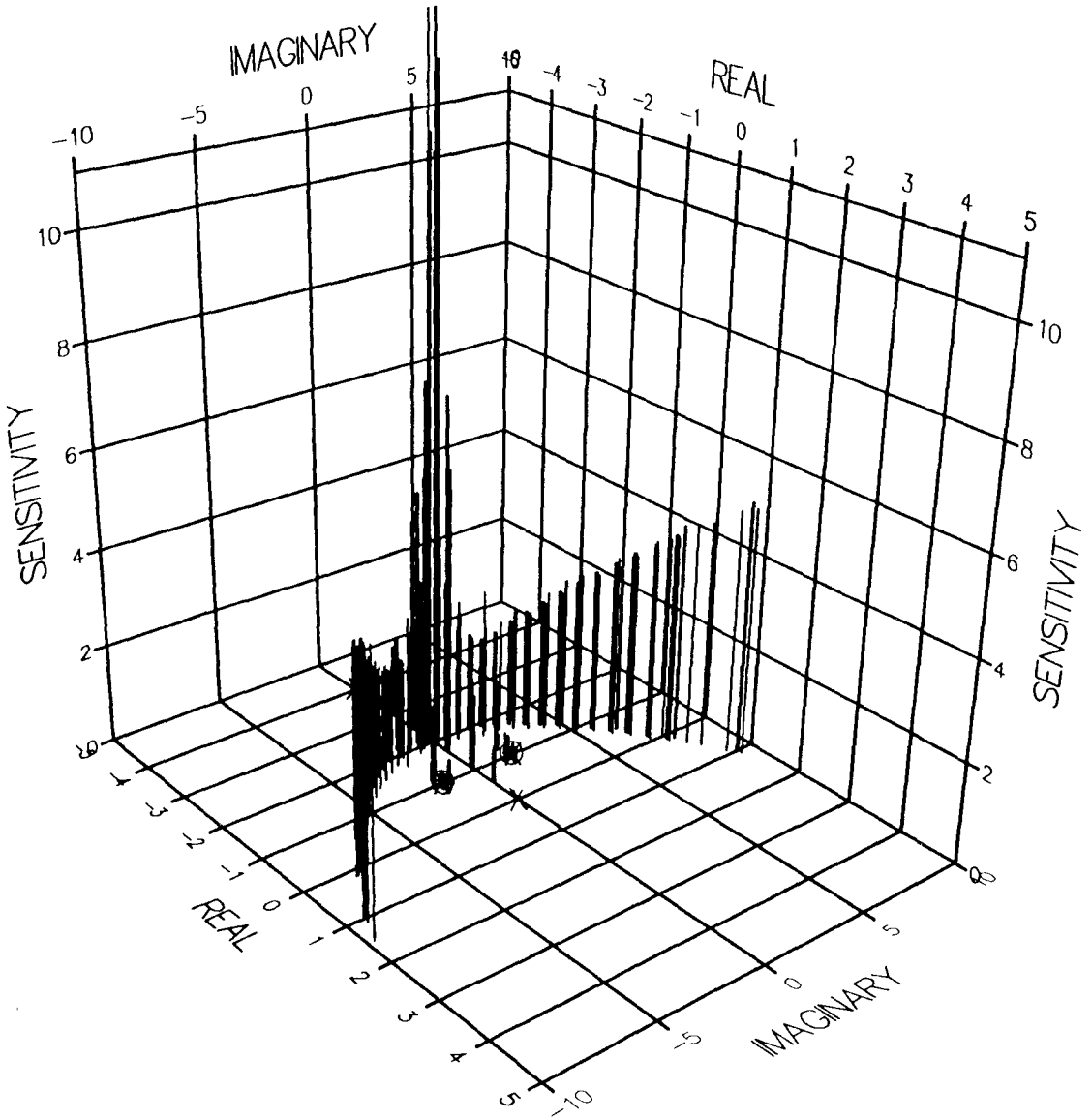


Figure 6.15 Boundary Surface Of The Root-Sensitivity Profiles Of The Fixed-Configuration Optimum Control System For Plant Of Case-Study "A" With Variations Of The Parameters "ξ_1", "ω_1", "K_p" While The Delay Is Kept Constant At 0.05 sec.

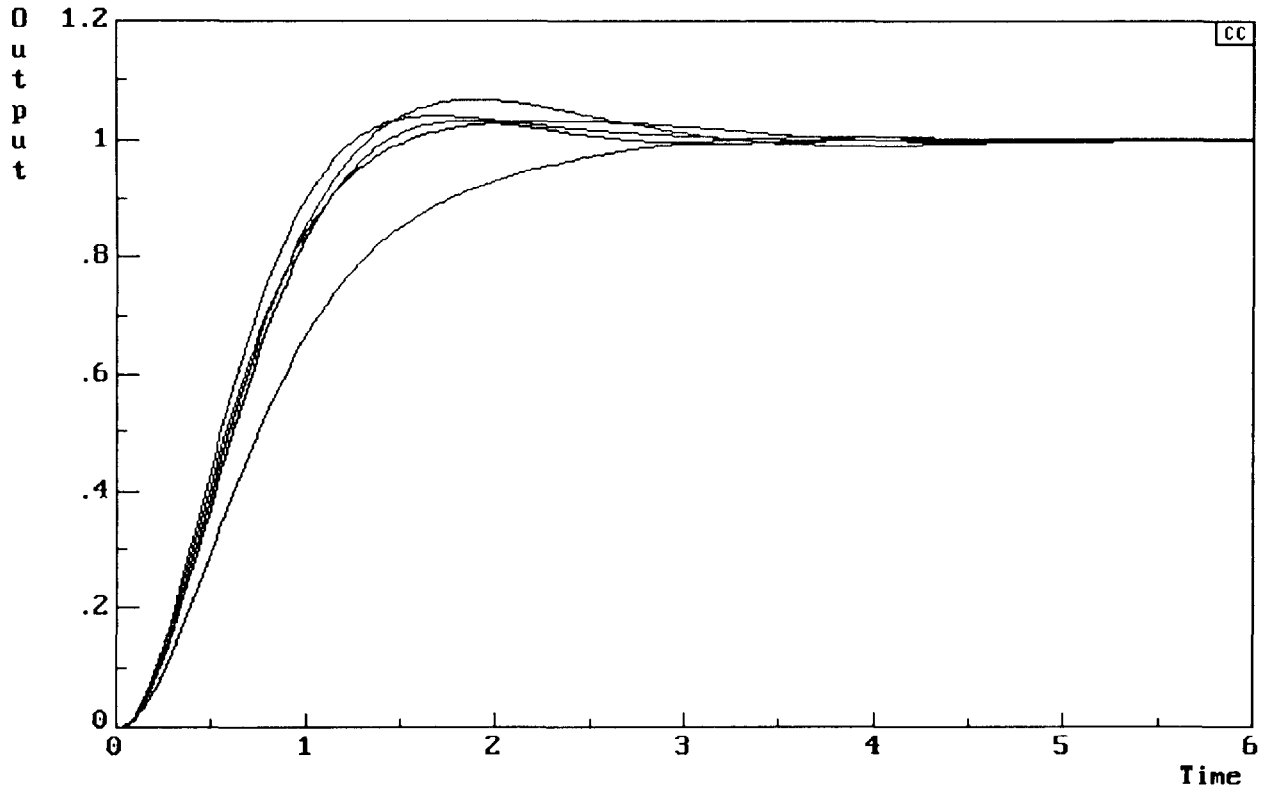


Figure 6.16 Family Of Transient Responses Of The Fixed-Configuration Optimum Control System For Plant Of Case-Study "A" With Variations Of The Parameters "ξ_1", "ω_1" While "K_p" Is Maintained At 5% Below Its Nominal Value And The Delay Is Kept Constant At 0.0 sec.

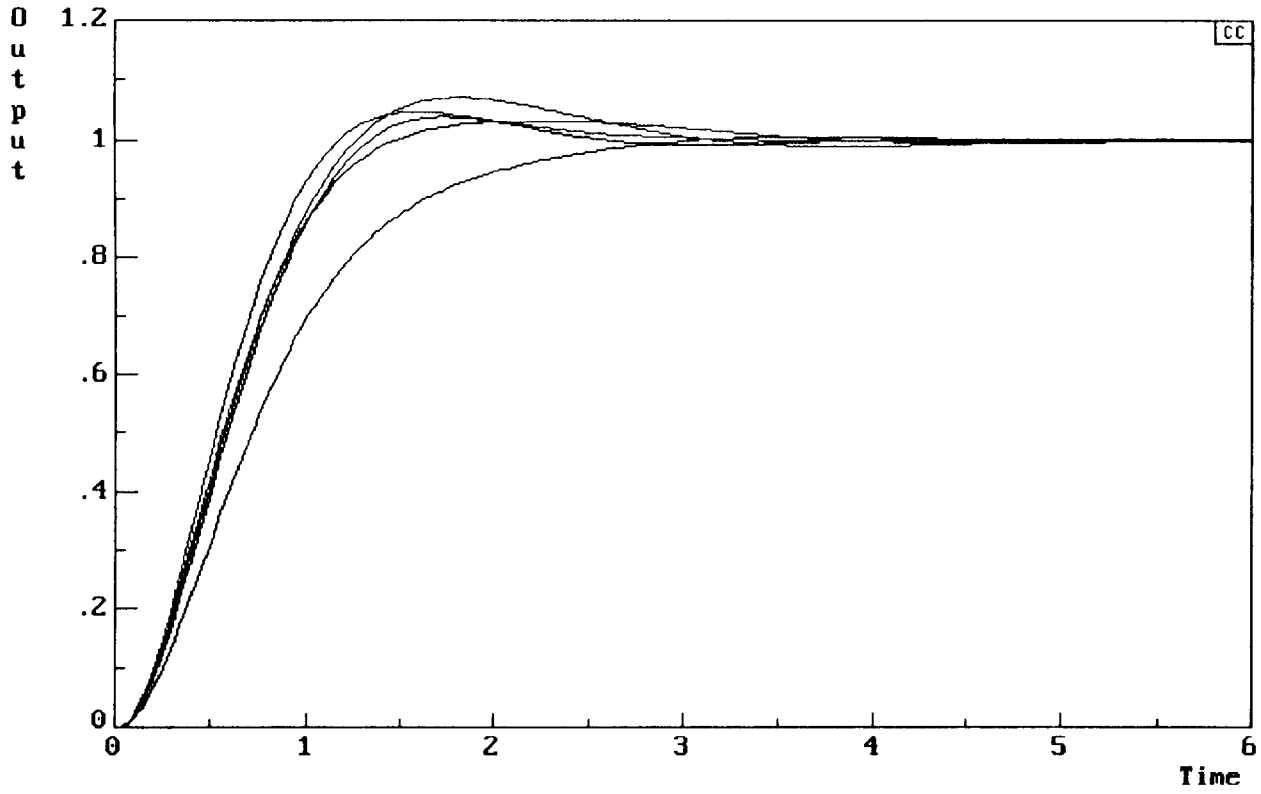


Figure 6.17 Family Of Transient Responses Of The Fixed-Configuration Optimum Control System For Plant Of Case-Study "A" With Variations Of The Parameters "ξ_1", "ω_1" While "K_p" Is Maintained At Its Nominal Value And The Delay Is Kept Constant At 0.0 sec.

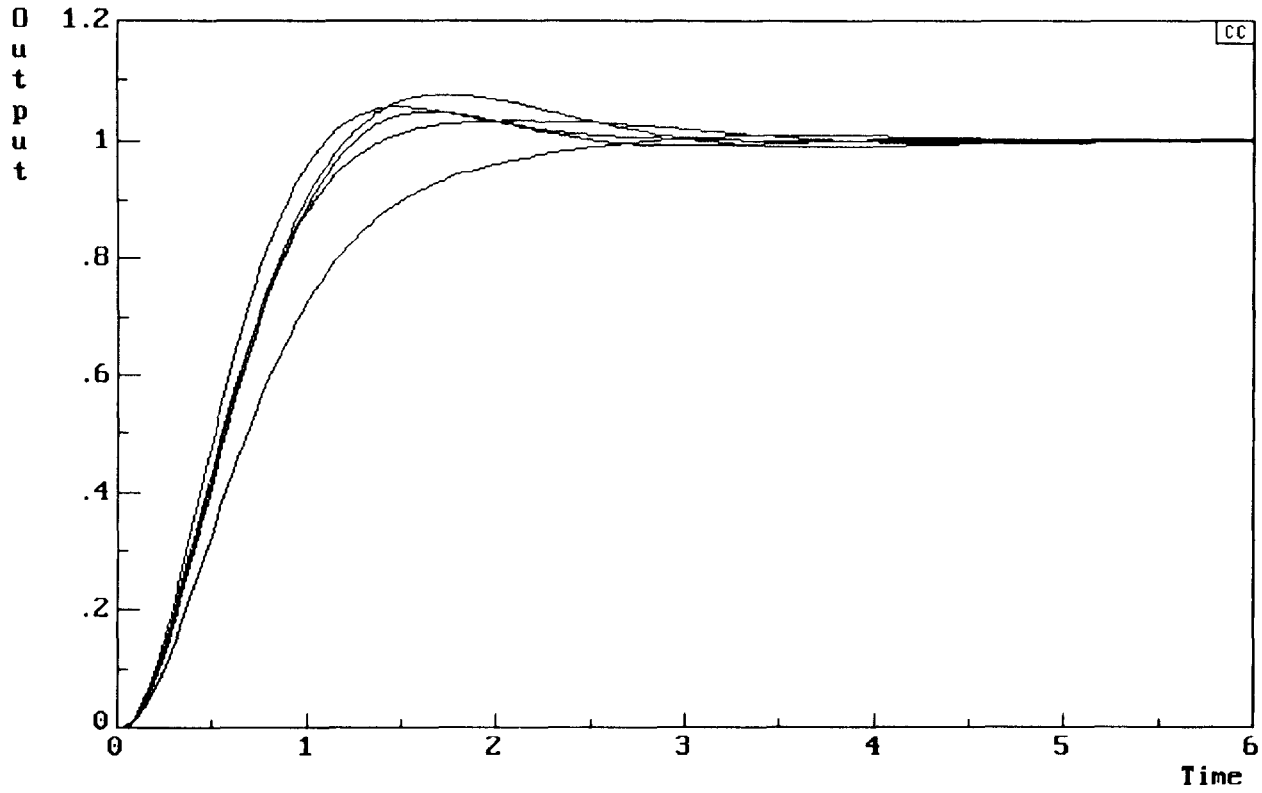


Figure 6.18 Family Of Transient Responses Of The Fixed-Configuration Optimum Control System For Plant Of Case-Study "A" With Variations Of The Parameters "ξ_1", "ω_1" While "K_p" Is Maintained At 5% Above Its Nominal Value And The Delay Is Kept Constant At 0.0 sec.

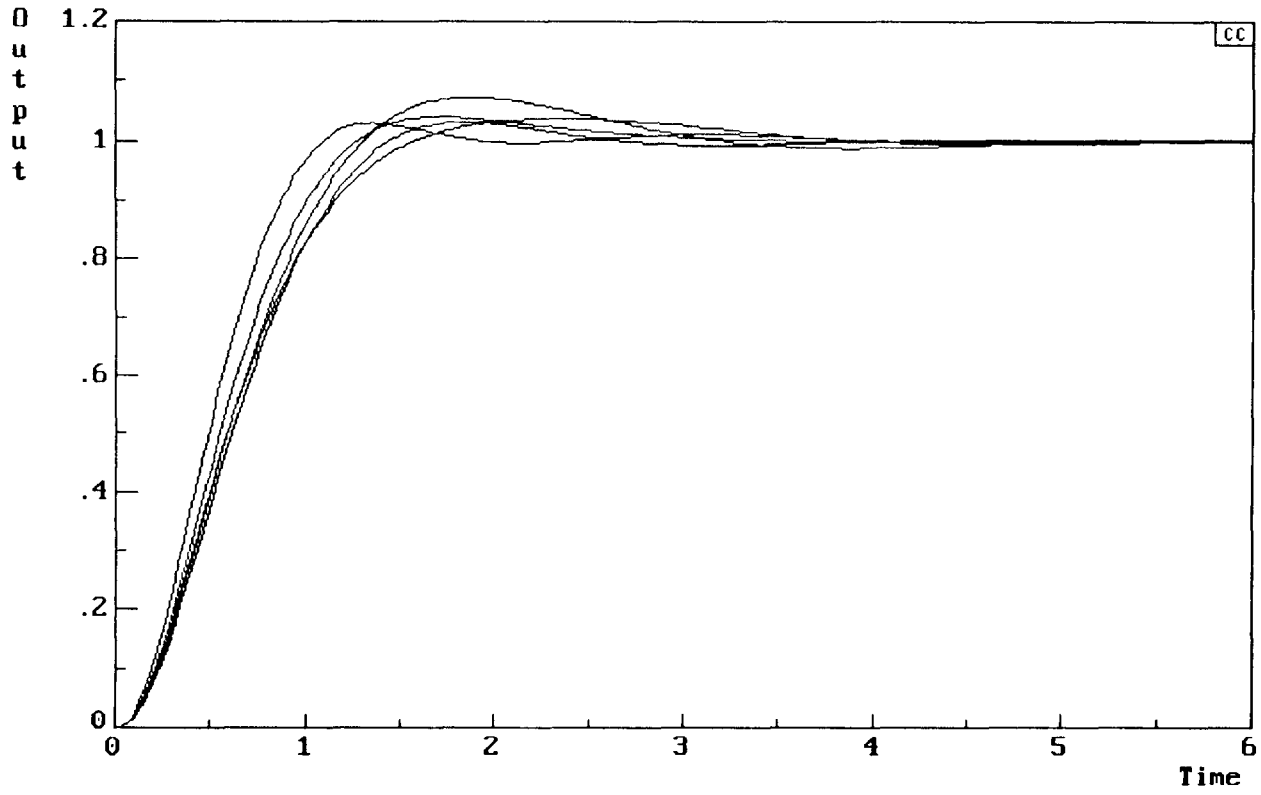


Figure 6.19 Family Of Transient Responses Of The Fixed-Configuration Optimum Control System For Plant Of Case-Study "A" With Variations Of The Parameters "ξ_1", "ω_1" While "K_p" Is Maintained At 5% Below Its Nominal Value And The Delay Is Kept Constant At 0.05 sec.

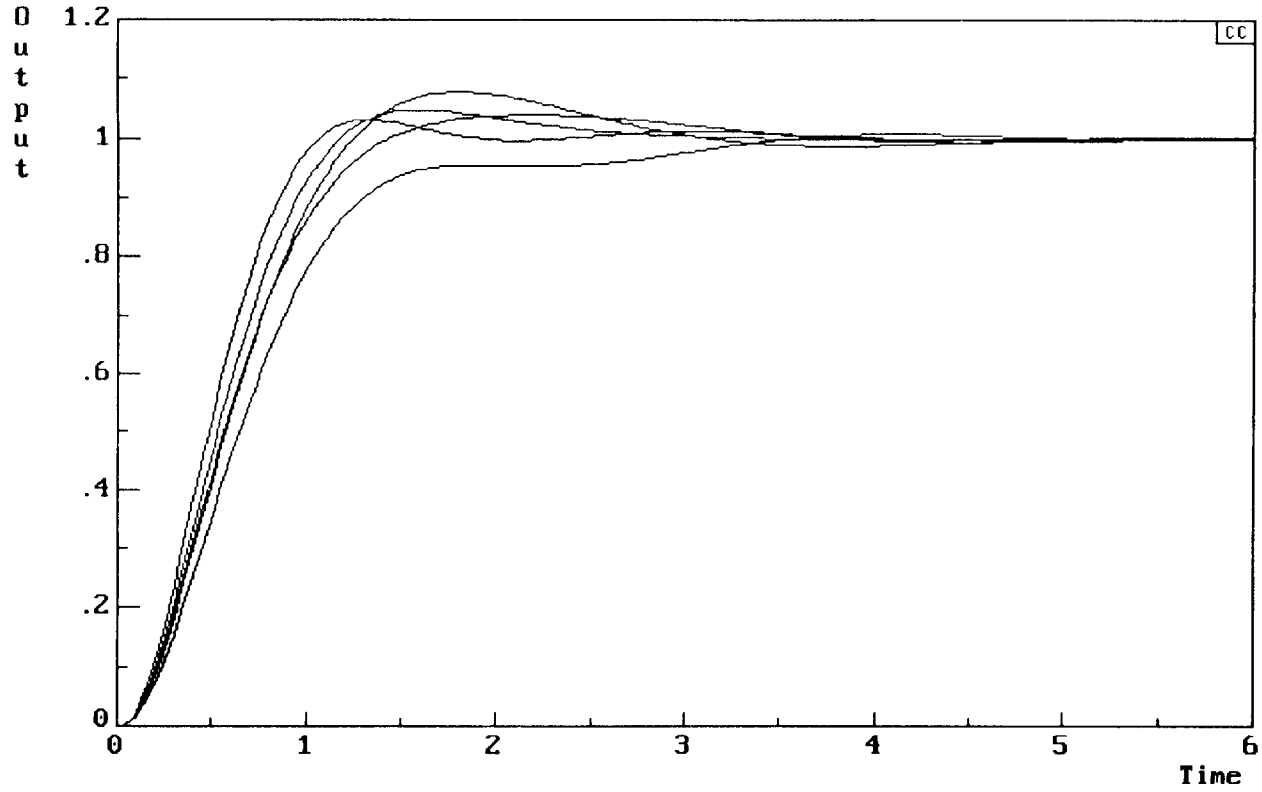


Figure 6.20 Family Of Transient Responses Of The Fixed-Configuration Optimum Control System For Plant Of Case-Study "A" With Variations Of The Parameters "ξ_1", "ω_1" While "K_p" Is Maintained At Its Nominal Value And The Delay Is Kept Constant At 0.05 sec.

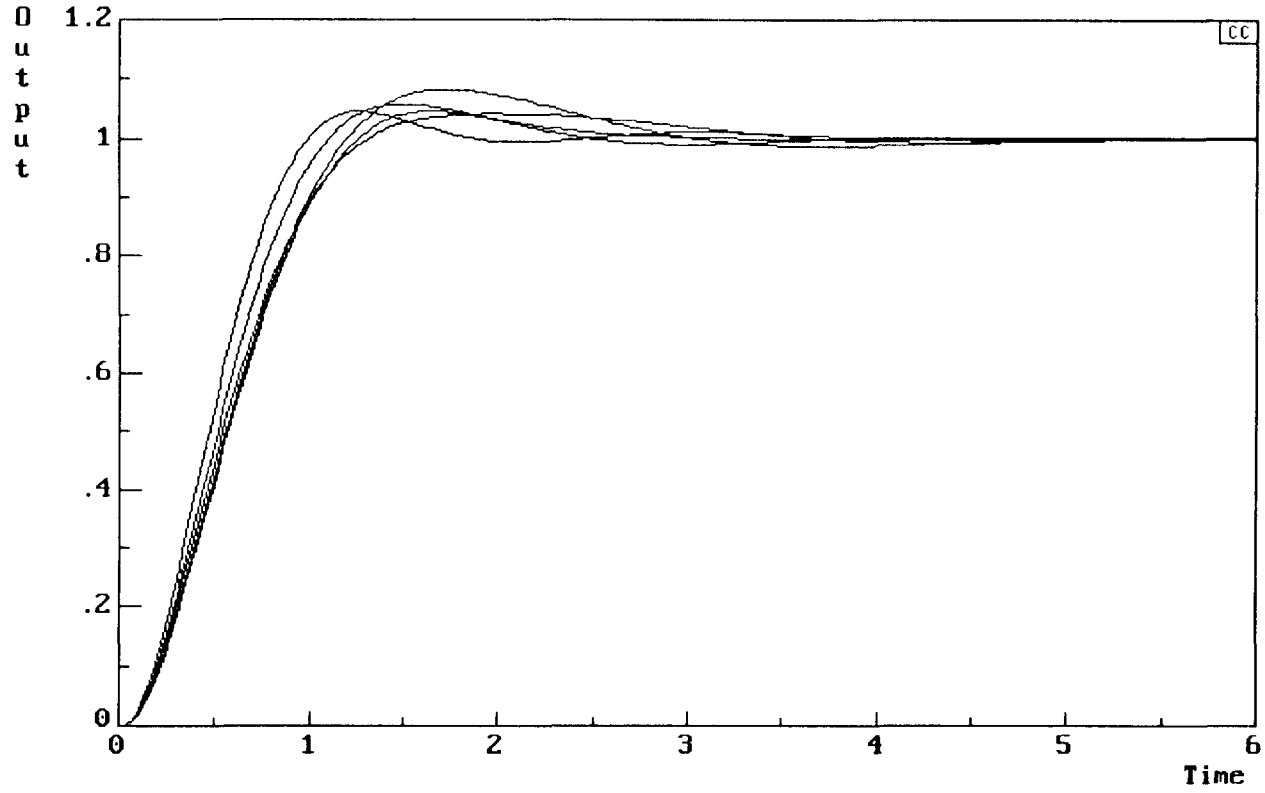


Figure 6.21 Family Of Transient Responses Of The Fixed-Configuration Optimum Control System For Plant Of Case-Study "A" With Variations Of The Parameters "ξ_1", "ω_1" While "K_p" Is Maintained At 5% Above Its Nominal Value And The Delay Is Kept Constant At 0.05 sec.

6.4 CAD Procedure For Adaptive Controller Design

In the case study of the previous section, it has been shown that a fixed-configuration controller, designed to optimise the system's dominant root sensitivities, has narrowly failed to meet all the performance criteria in a relatively small region within the plant's parametric-variation ranges. Although this may be tolerated if the performance criteria can be slightly relaxed, the degree of the system's sensitivities to plant's parametric variations remains too high to be tolerated. These analyses indicate that improvements, particularly to the system's integrity, are likely to be achieved only by an adaptive controller.

Adapting the general structure of the fixed-configuration optimum controller "$G_{CF}(S)$" of equation 5.12, except for allowing its complex zeros to be adaptively adjusted, yields the transfer function:

$$G_{CA}(S) = \frac{K_{CA}\,(S^2 + 2\xi_A\omega_A\,S + \omega_A{}^2)}{S\,(S+40.0)}\,,$$

(6.1)

where "ξ_A" and "ω_A" are control parameters assumed to be adjusted automatically in addition to "K_{CA}" to optimise the sensitivities of the system's dominant closed-loop poles.

Since variations of "ξ_A" and "ω_A" directly translate into variations of the controller's complex zeros in the S-plane, it is possible to adopt a strategy whereby **these zeros continuously shadow the migrations of the plant's complex poles** (see case study A in section 6.2) so that they remain to the east of their respective plant's poles at a distance of at least 0.05. This finite distance is an indirect representation of inaccuracies and non-linearities inherent in transducers, finite resolution of the

computations, noise,....etc. inevitably present in all real plants which render exact cancellation of the plant's poles practically impossible (refer to section 5.4).

Using the sensitivity-based algorithm as a CAD system and keeping in mind the comments in section 6.2 regarding the diverse on-screen facilities, relevant frames of the boundary surface diagrams were captured for extensive off-line evaluations. Snapshots of the resulting boundary surfaces of the root contour diagrams of the optimum autopilot design are shown in figures 6.22 and 6.24- superimposed onto the performance specifications loci as a background- for plant delays of 0.0 and 0.05 seconds respectively. Whereas figures 6.23 and 6.25 show snapshots of the boundary surfaces of the corresponding root sensitivities profiles.

Significantly, the adaptive controller has reduced the thickness of the boundary surface graphs to almost a single system of loci which imply considerable improvement to the control system robustness. The results summarised in tables 6.3A and 6.3B for plant delays of 0.0 and 0.05 seconds, respectively, show **the ability of the adaptive controller to fine tune the system's performance to operate within a very narrow zone close to its optimum despite the relatively wide variations of the plant parameters and in spite of deliberately disabling the automatic gain compensation of the controller for the plant's gain variations.** These results show not only that the constrained control system satisfies all the performance targets (outlined in section 5.2.2), but also that the system is at least comparable, if not superior to , the control system designed for the deterministic plant across all the performance indices.

A further analysis of the results have been conducted in a similar way as in section 6.3 in order to derive quantitative measures of the integrity of the system's behaviour. For this purpose, the constrained adaptive control system has been simulated (using Program CC version 4, see reference [57]) with the values of controller parameters

Table 6.3A Results Of The Gain-Constrained Adaptive Control System's Design For A Plant With Parametric Variations, Gain Variations Of ± 5%, And With A Delay Of 0.0 Sec.(Figures Rounded To Four Decimal Places)

Plant Gain Variation*	Plant's Complex Poles Migrations Around The Perimeter Of Their Respective Fuzzy Discs	Sensitivity Of The Dominant Closed-Loop poles	Bandwidth (rad. s^{-1})	Loop Gain ($K_P K_{CA}$) (rad. s^{-1})	Attenuation Required Outside The Loop For Unity d.c. Gain	Velocity Error Constant K_V (rad. s^{-1})	Steady-State Error For A Unit-Ramp Input	Time-Domain Performance For A Unit-Step Target Input		
								Percent Overshoot	Settling Time (sec.)	Rise Time (sec.)
Lower Threshold	Centre	1.8939	2.8702	306.7360	1.0000	1.8669	0.5356	3.9890	2.3635	0.7922
	East	1.9167	2.8638	307.0512	1.0000	1.8749	0.5333	3.9890	2.3702	0.7922
	North	1.9011	2.9166	317.1894	1.0000	1.9380	0.5160	4.0430	2.3714	0.7606
	West	1.8655	2.8678	304.1660	1.0000	1.8524	0.5398	4.7340	2.3555	0.7922
	South	1.8894	2.8239	296.4448	1.0000	1.8034	0.5545	3.4040	2.3560	0.8239
Optimum	Centre	1.8889	2.9474	324.2554	1.0000	1.9736	0.5067	4.6810	2.3636	0.7500
	East	1.9112	2.9408	324.3746	1.0000	1.9807	0.5049	4.6810	2.3708	0.7394
	North	1.8959	2.9953	335.2367	1.0000	2.0482	0.4882	5.4790	2.3708	0.7183
	West	1.8607	2.9452	321.8123	1.0000	1.9599	0.5102	4.6810	2.3548	0.7482
	South	1.8840	2.8996	313.4198	1.0000	1.9066	0.5245	4.0430	2.3566	0.7711
Upper Threshold	Centre	1.8947	3.0253	342.7698	1.0000	1.9767	0.5059	5.4790	2.3656	0.6972
	East	1.9150	3.0185	342.6823	1.0000	2.0925	0.4779	5.4790	2.3730	0.6972
	North	1.9017	3.0745	354.3062	1.0000	2.1648	0.4619	6.2770	2.3723	0.6761
	West	1.8687	3.0229	340.4631	1.0000	2.0735	0.4823	5.4790	2.3565	0.7130
	South	1.8892	2.9760	331.3657	1.0000	2.0158	0.4961	5.0000	2.3590	0.7289

* Refers To The Upper & The Lower Boundaries Of The Plant-Gain Variations (Of ± 5%) Around Its Nominal Value.

Table 6.3B Results Of The Gain-Constrained Adaptive Control System's Design For A Plant With Parametric Variations, Gain Variations Of ± 5%, And With A Delay Of 0.05 Sec.(Figures Rounded To Four Decimal Places)

Plant Gain Variation*	Plant's Complex Poles Migrations Around The Perimeter Of Their Respective Fuzzy Discs	Sensitivity Of The Dominant Closed-Loop Poles	Bandwidth (rad. s^{-1})	Loop Gain ($K_P K_C$) (rad. s^{-1})	Attenuation Required Outside The Loop For Unity d.c. Gain	Velocity Error Constant K_V (rad. s^{-1})	Steady-State Error For A Unit-Ramp Input	Time-Domain Performance For A Unit-Step Target Input		
								Percent Overshoot	Settling Time (sec.)	Rise Time (sec.)
Lower Threshold	Centre	1.8860	2.8716	339.4047	0.8952	2.0658	0.4841	3.9870	2.3665	0.7922
	East	1.9146	2.8404	339.3717	0.8926	2.0722	0.4825	3.9890	2.3708	0.7922
	North	1.8919	2.9153	351.2486	0.8922	2.1461	0.4660	4.7340	2.3820	0.7711
	West	1.8595	2.8631	334.6024	0.8969	2.0378	0.4907	3.6170	2.3564	0.8099
	South	1.8825	2.8295	328.1057	0.8982	1.9960	0.5010	3.2450	2.3517	0.8294
Optimum	Centre	1.8809	2.9411	358.8757	0.8898	2.1843	0.4578	4.8360	2.3802	0.7500
	East	1.9081	2.9295	358.5464	0.8879	1.9807	0.5049	4.6810	2.3849	0.7500
	North	1.8861	2.9863	371.3380	0.8870	2.2688	0.4407	5.4790	2.3957	0.7183
	West	1.8523	2.9328	354.0941	0.8914	2.1565	0.4637	4.7340	2.3687	0.7570
	South	1.8775	2.8972	346.9639	0.8928	2.1107	0.4738	3.9896	2.3655	0.7817
Upper Threshold	Centre	1.8879	3.0109	379.4510	0.8849	2.3095	0.4330	5.7980	2.3970	0.6972
	East	1.9122	2.9992	378.8161	0.8826	2.3131	0.4323	5.8510	2.4018	0.7077
	North	1.8931	3.0576	392.5676	1.1336	2.3985	0.4169	6.5960	2.4126	0.6866
	West	1.8602	3.0025	374.7140	0.8863	2.2821	0.4382	5.4790	2.3847	0.7130
	South	1.8842	2.9656	366.8946	1.0344	2.2319	0.4480	5.1060	2.3822	0.7289

* Refers To The Upper & The Lower Boundaries Of The Plant-Gain Variations (Of ± 5%) Around Its Nominal Value.

"ξ_A", "ω_A" and "K_{CA}" derived from the CAD system at the optimum operating conditions which correspond to the captured data frames given in table 6.3A and 6.3B. Figures 6.26 to 6.28 show three sets of the system transient responses to a unit-step pitch target angle for a zero plant delay, while figures 6.29 to 6.31 show the same for 0.05 seconds plant delay. Both are classified according to plant gain variations of $\pm$ 5% of its nominal value. Each set consists of five responses; four of which correspond to the migrations of the plant's complex poles around the perimeter of their fuzzy discs, and the fifth corresponds to the plant's complex poles located at the centre of their respective fuzzy discs (identical to the deterministic case).

These graphs, apart from corroborating the results of the time-domain performance indices predicted by the developed CAD system (as given in columns 8,9 and 10 of tables 6.3A and 6.3B), they also show very narrow variations relative to the case of the optimum control of the deterministic plant. The sensitivity of the adaptive control system to plant's parametric variations can be calculated in terms of the mean deviation and the mean variance of its transient responses relative to that of the optimum control system of the deterministic plant. These sensitivity measures are summarised in tables 6.4A and 6.4B for plant's delays of 0.0 and 0.05 seconds respectively. The results in each table are divided into three sets classified according to the plant gain variations of $\pm$ 5% around its nominal value. Each set contains four comparisons which correspond to the migrations of the plant's complex poles around the perimeter of their respective fuzzy discs in the order: East, North, West and South.

These results show that the mean deviation does not exceed 0.0041 and that the mean variance does not exceed 0.00005. Compared with the figures obtained for the optimum control system for the deterministic plant, the constrained adaptive control system show superior robust qualities for plants with uncertainties. Without the gain restriction referred to above, the adaptive control system is expected to exhibit even better overall performance and robust qualities.

Table 6.4A Quality Assurance Measures Of The Gain-Constrained Adaptive Control System In Terms Of Transient Response Sensitivity To Plant Parametric Variations And To Plant Gain Variations Of ± 5%, For 0.0 Sec. Delay.

Plant Gain Variations*	Plant's Complex Poles Migrations Around The Perimeter Of Their Respective Fuzzy Discs	Transient Response Sensitivity To Plant Parametric Variations Relative To Its Deterministic Case.	
		Mean Deviation	Mean Variance
Lower Threshold	East	0.000509	0.000000
	North	0.004019	0.000049
	West	0.001009	0.000002
	South	**0.004060**	**0.000050**
Optimum	East	0.000520	0.000000
	North	0.003824	0.000047
	West	0.000922	0.000002
	South	0.003929	0.000049
Upper Threshold	East	0.000573	0.000001
	North	0.003871	0.000042
	West	0.000820	0.000002
	South	0.003773	0.000047

* Refers To The Upper & The Lower Boundaries Of The Plant-Gain Variations (Of ± 5%) Around Its Nominal Value.

Table 6.4B Quality Assurance Measures Of The Gain-Constrained Adaptive Control System In Terms Of Transient Response Sensitivity To Plant Parametric Variations And To Plant Gain Variations Of ± 5%, For 0.05 Sec. Delay.

Plant Gain Variations*	Plant's Complex Poles Migrations Around The Perimeter Of Their Respective Fuzzy Discs	Transient Response Sensitivity To Plant Parametric Variations Relative To Its Deterministic Case.	
		Mean Deviation	Mean Variance
Lower Threshold	East	0.000953	0.0000001
	North	**0.004118**	**0.000050**
	West	0.001632	0.000007
	South	0.004094	0.000049
Optimum	East	0.000946	0.000001
	North	0.004005	0.000050
	West	0.001494	0.000006
	South	0.003973	0.000049
Upper Threshold	East	0.001095	0.000002
	North	0.003903	0.000049
	West	0.001365	0.000005
	South	0.003863	0.000048

* Refers To The Upper & The Lower Boundaries Of The Plant-Gain Variations (Of ± 5%) Around Its Nominal Value.

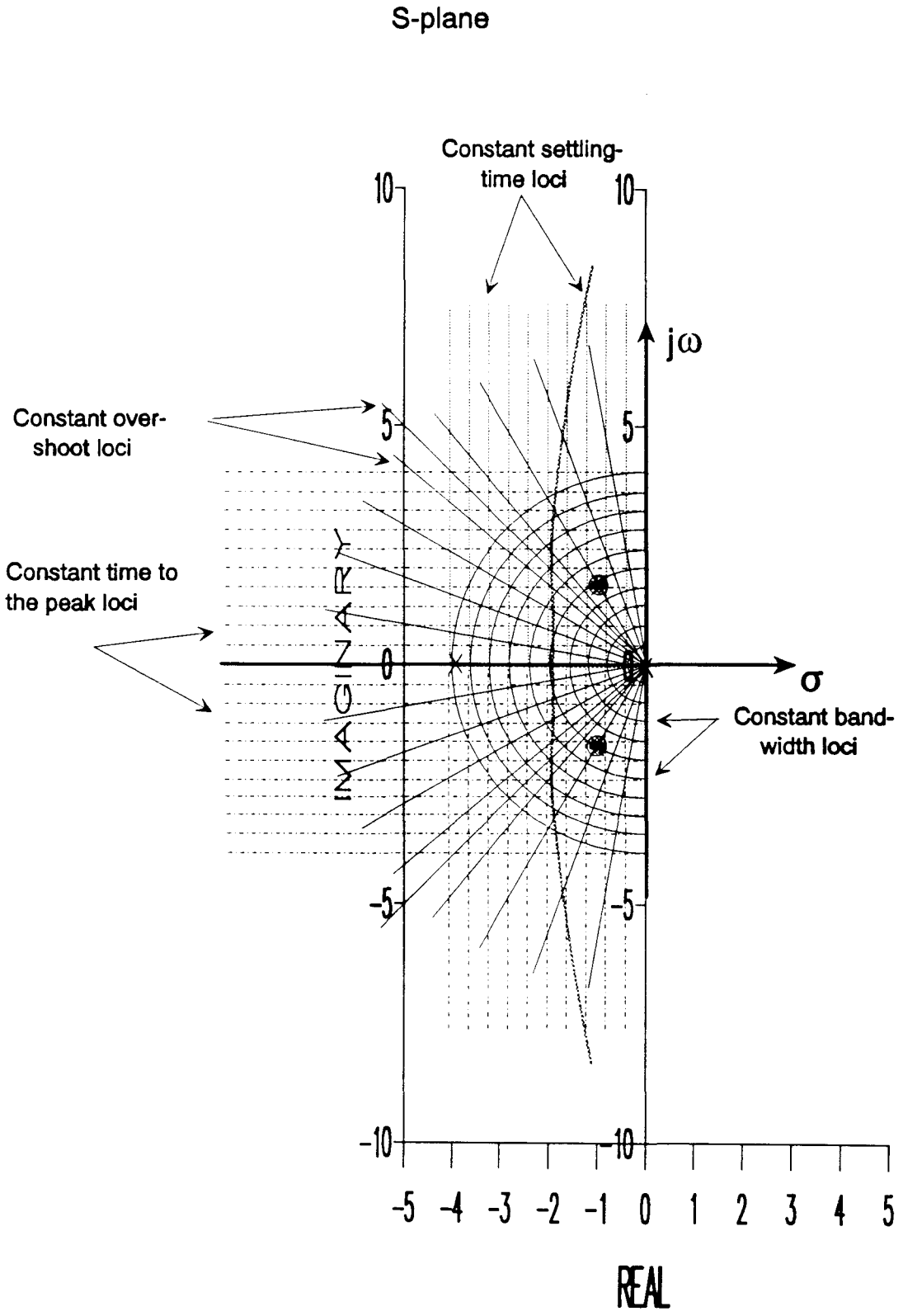


Figure 6.22 Boundary Surface Of The Root-Contour Diagrams Of The Gain-Constrained Adaptive Control System For Plant Of Case-Study "A" With Variations Of The Parameters "ξ_1", "ω_1", "K_p" While The Delay Is Kept Constant At 0.0 sec. (Performance Specifications Loci Are Superimposed)

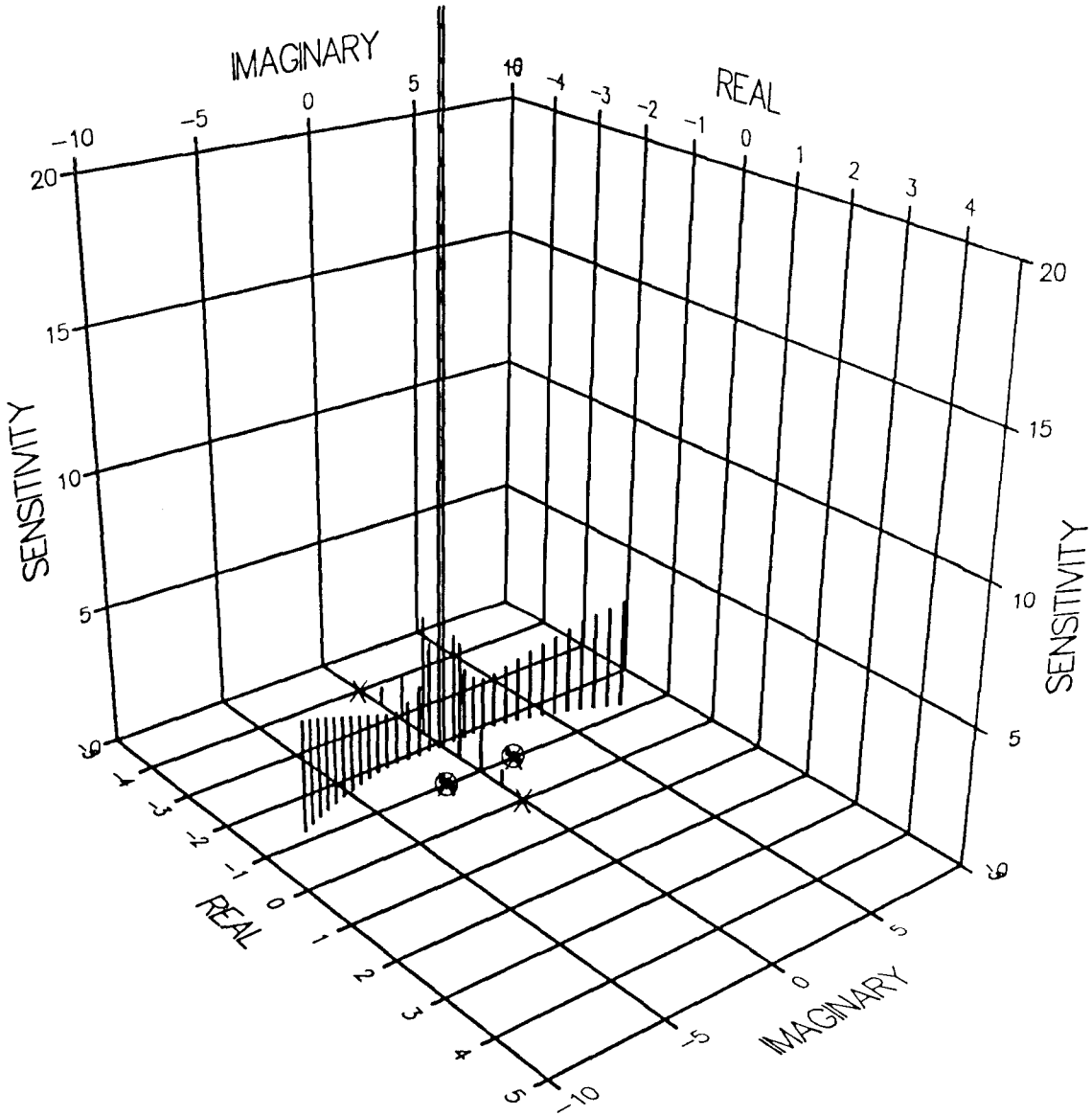


Figure 6.23 Boundary Surface Of The Root-Sensitivity Profiles Of The Gain-Constrained Adaptive Control System For Plant Of Case-Study "A" With Variations Of The Parameters "ξ_1", "ω_1", "K_p" While The Delay Is Kept Constant At 0.0 sec.

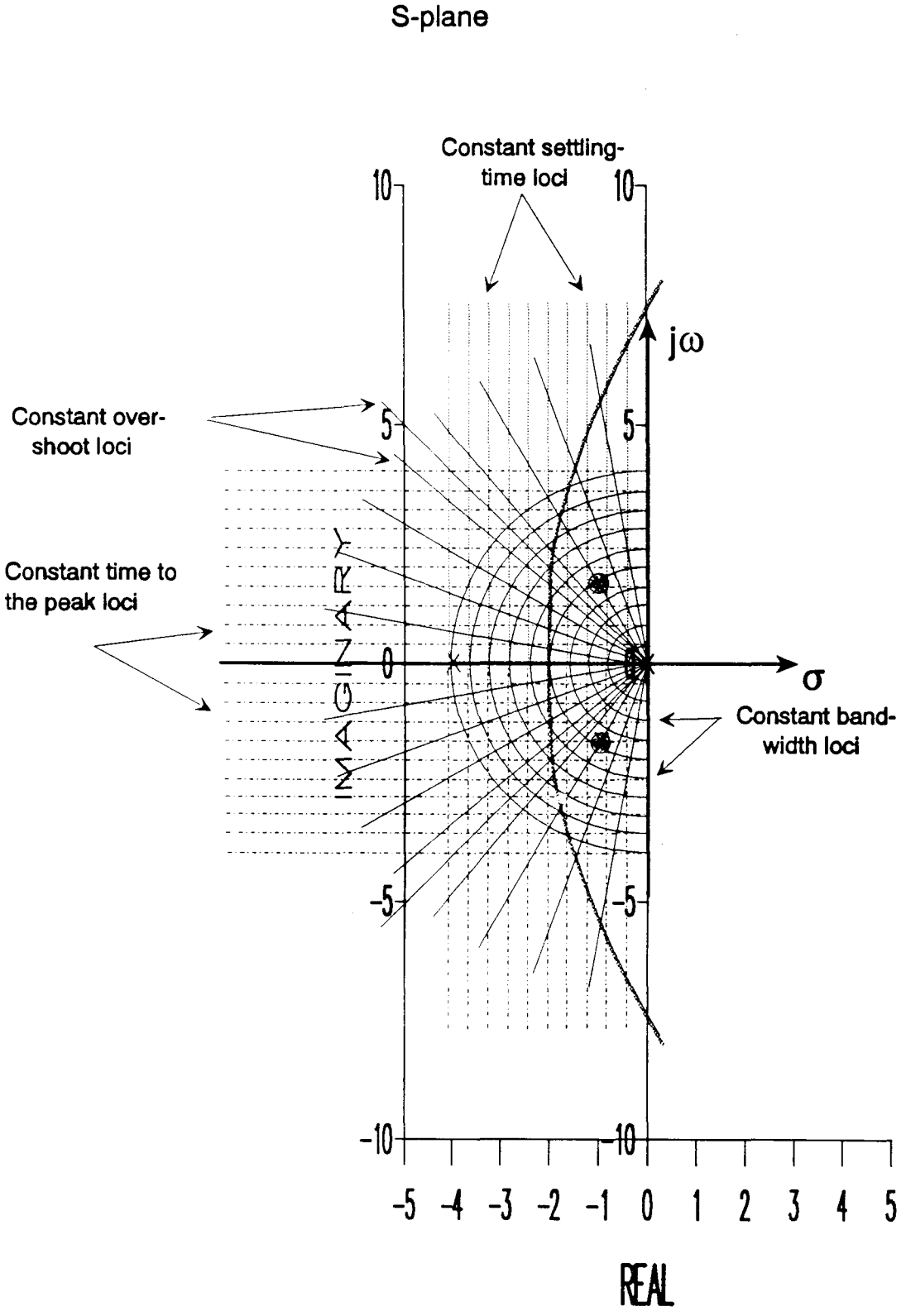


Figure 6.24 Boundary Surface Of The Root-Contour Diagrams Of The Gain-Constrained Adaptive Control System For Plant Of Case-Study "A" With Variations Of The Parameters "ξ_1", "ω_1", "K_p" While The Delay Is Kept Constant At 0.05 sec. (Performance Specifications Loci Are Superimposed)

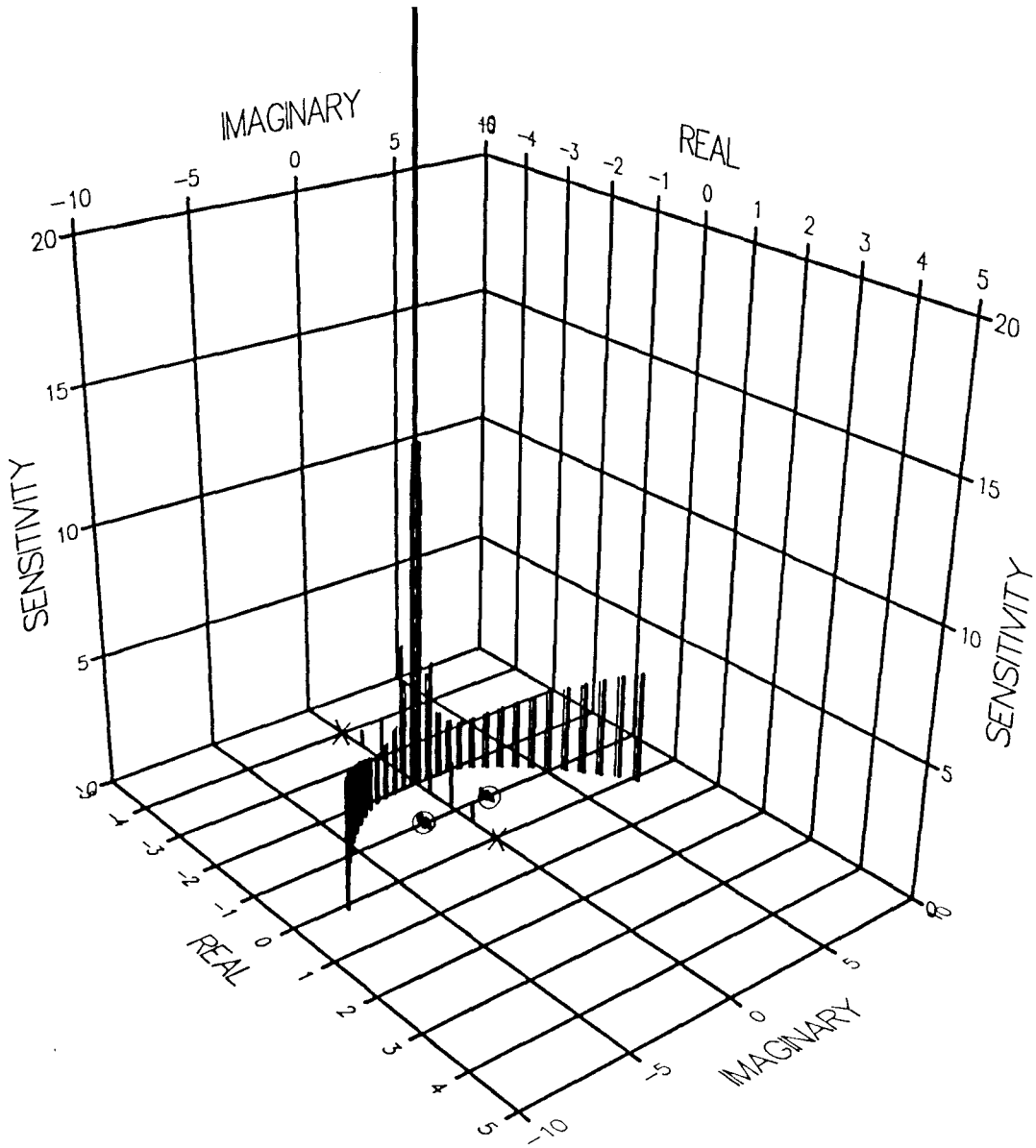


Figure 6.25 Boundary Surface Of The Root-Sensitivity Profiles Of The Gain-Constrained Adaptive Control System For Plant Of Case-Study "A" With Variations Of The Parameters "ξ_1", "ω_1", "K_p" While The Delay Is Kept Constant At 0.05 sec.

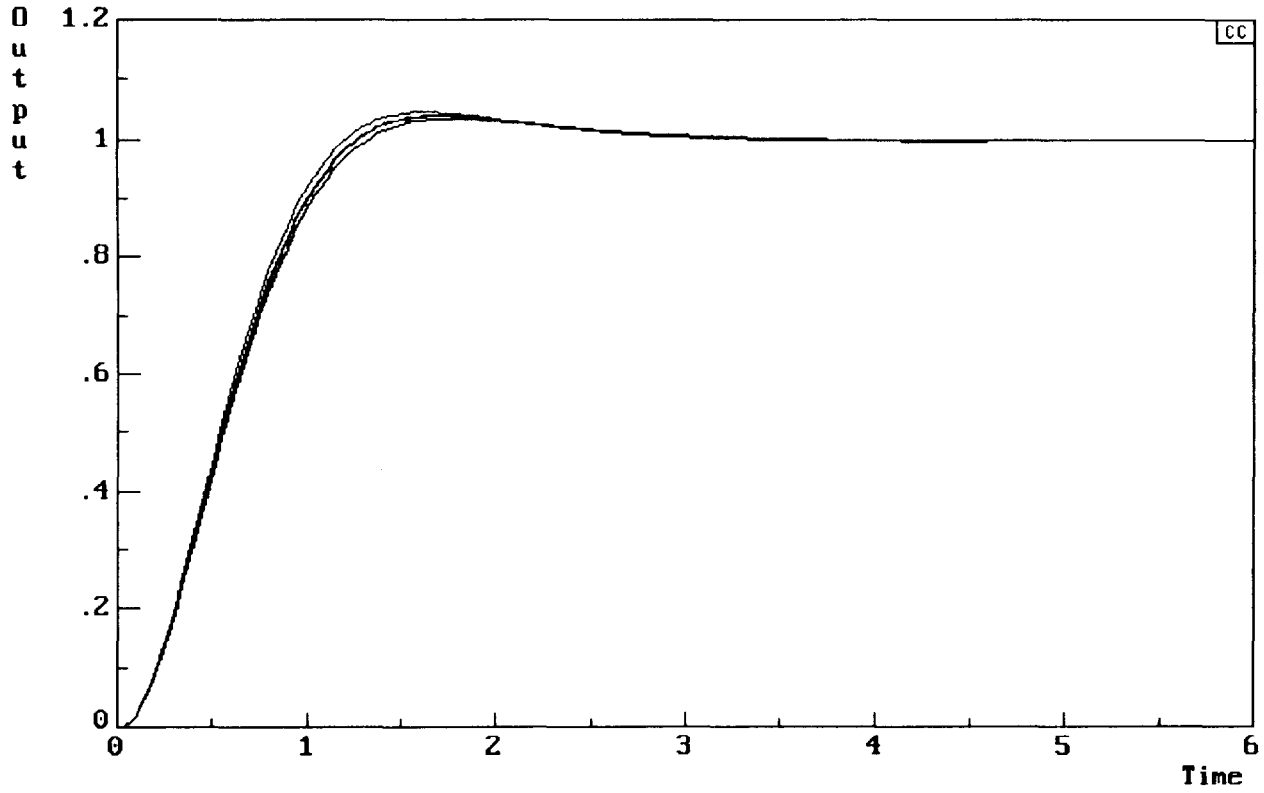


Figure 6.26 Family Of Transient Responses Of The Gain-Constrained Adaptive Control System For Plant Of Case-Study "A" With Variations Of The Parameters "ξ_1", "ω_1" While "K_p" Is Maintained At 5% Below Its Nominal Value And The Delay Is Kept Constant At 0.0 sec.

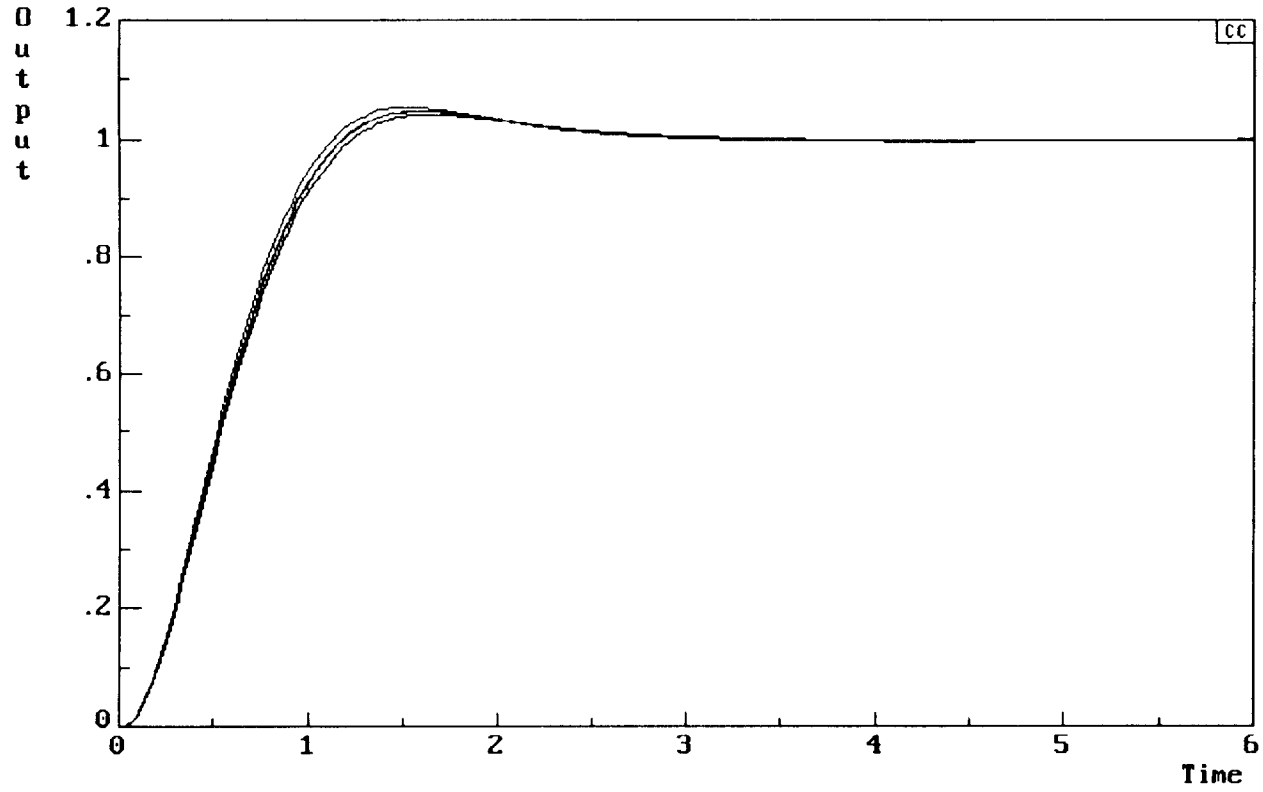


Figure 6.27 Family Of Transient Responses Of The Gain-Constrained Adaptive Control System For Plant Of Case-Study "A" With Variations Of The Parameters "ξ_1", "ω_1" While "K_p" Is Maintained At Its Nominal Value And The Delay Is Kept Constant At 0.0 sec.

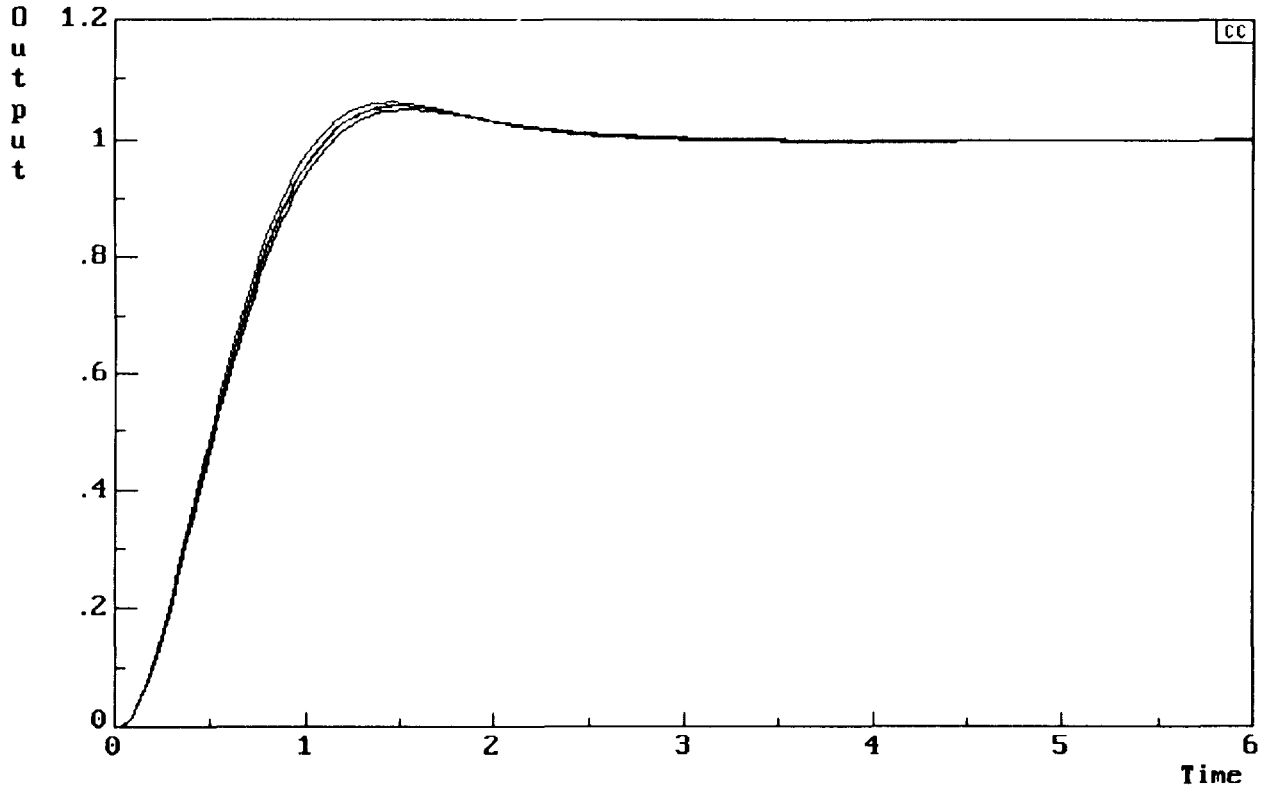


Figure 6.28 Family Of Transient Responses Of The Gain-Constrained Adaptive Control System For Plant Of Case-Study "A" With Variations Of The Parameters "ξ_1", "ω_1" While "K_p" Is Maintained At 5% Above Its Nominal Value And The Delay Is Kept Constant At 0.0 sec.

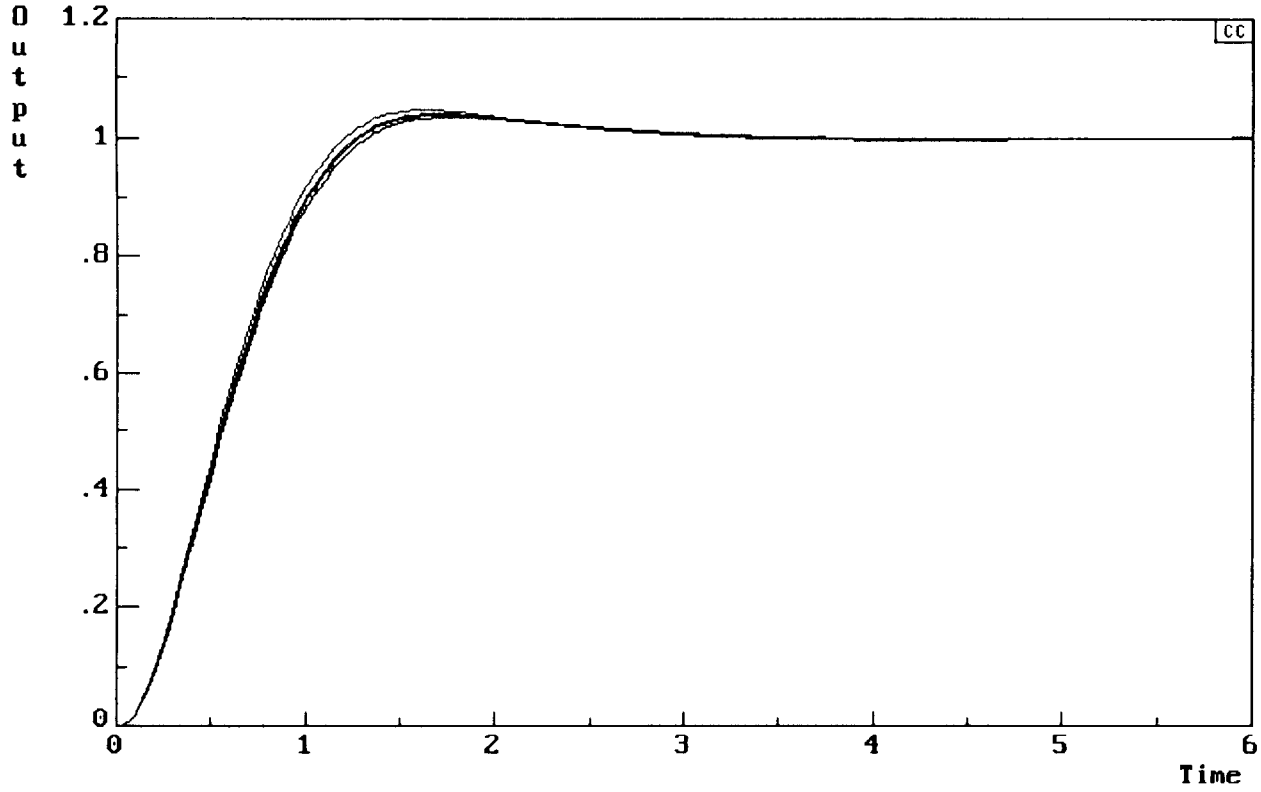


Figure 6.29 Family Of Transient Responses Of The Gain-Constrained Adaptive Control System For Plant Of Case-Study "A" With Variations Of The Parameters "ξ_1", "ω_1" While "K_p" Is Maintained At 5% Below Its Nominal Value And The Delay Is Kept Constant At 0.05 sec.

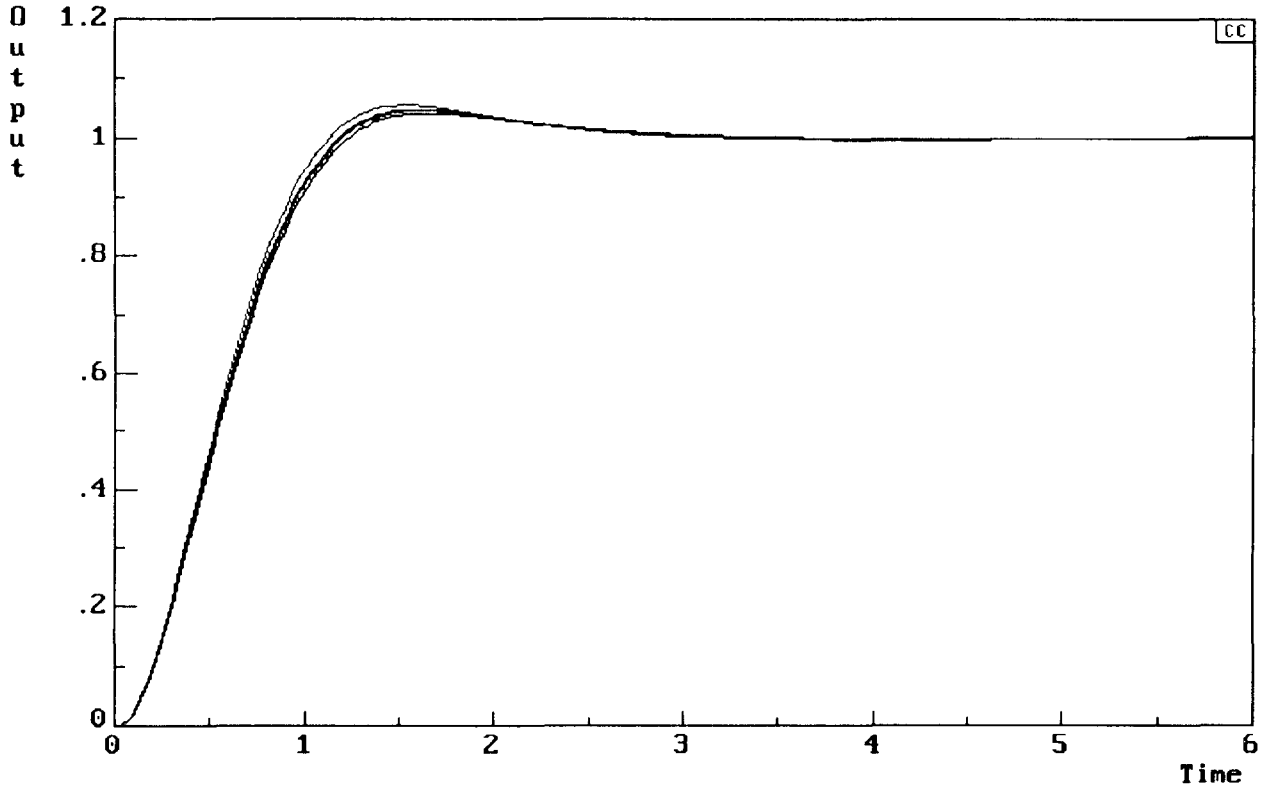


Figure 6.30 Family Of Transient Responses Of The Gain-Constrained Adaptive Control System For Plant Of Case-Study "A" With Variations Of The Parameters "ξ_1", "ω_1" While "K_p" Is Maintained At Its Nominal Value And The Delay Is Kept Constant At 0.05 sec.

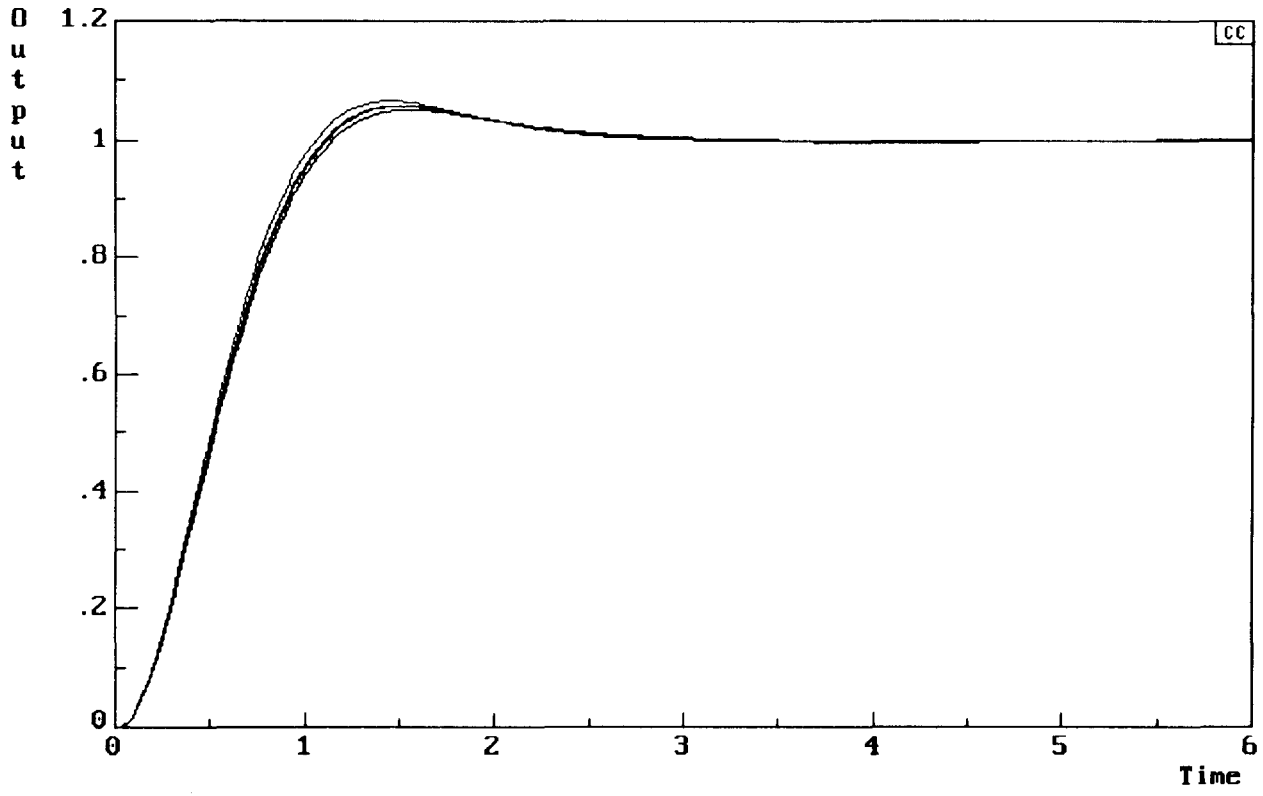


Figure 6.31 Family Of Transient Responses Of The Gain-Constrained Adaptive Control System For Plant Of Case-Study "A" With Variations Of The Parameters "ξ_1", "ω_1" While "K_p" Is Maintained At 5% Above Its Nominal Value And The Delay Is Kept Constant At 0.05 sec.

6.5 Conclusions

The studies and analyses in this chapter have shown that the sensitivity-based algorithm is a useful tool for modelling plants with uncertainties. It has also shown its diverse facilities as a powerful CAD system for both fixed-configuration controllers and adaptive controllers which optimise not only the time-domain performance indices, but also the robustness qualities of the control systems operating in real environments. The real-time computations of the boundary surfaces of the root contour and the root sensitivity loci, accompanied with fast on-screen interactive graphics and easy access to evaluations procedures, represent key developments to assist a host of entwined design decisions. With optimised software programming and further enhancements to its inbuilt automated procedures, the algorithm has the potential of being used on-line as self-tuned controllers to diverse real-life plants.

Chapter 7

Application Of The Sensitivity-Based Algorithm To The Design Of Reliable/Fault Tolerant Control Systems

7.1 Introduction

With the ever growing development of highly sophisticated systems, reliable/fault tolerant systems designs are becoming mandatory particularly when health, safety and adverse environmental effects are implicated. In many applications, the failure of some sub-systems or components may not necessarily cause a complete system breakdown, but rather degrade invariably its performance. Therefore, building systems that can reliably sustain a certain performance level throughout their expected lives has been recently perceived as one of the top design priorities. Avionics and communications are a few examples where reliable and fault-tolerant designs are assuming essential features.

After a brief review of recent work on reliability, fault tolerance and fault diagnoses, this chapter presents an approach to the simulation of a common class of fault in real plants using the sensitivity-based algorithm. With the help of the same algorithm as a CAD system, an adaptive controller is then designed for the autopilot case study, considered in the previous two chapters, with the objective of achieving a control system that is tolerant to a relatively wide range of plant faults.

7.1.1 Reliability

Reliability may be defined as the probability that a system operates without interruption during an interval of interest under specified condition (Barlow & Proshan[61]). Reliability can be extended to include several levels of system performance to allow analysis of degradable systems. A performance-oriented extension of reliability is to replace a single acceptable level of operation by a set of performance-level. This approach is used for evaluating network performance and reliability in (Aggarwal et. al.[62], Levy & Wirth[63], Tortorella[64]).

7.1.2. Fault tolerance

Fault tolerance implies the ability of the system to continue operation and provide services despite the failure of a few system components- Shieh et. al.[37]. Increasing the redundancy within the software is a common way to achieve fault tolerance. Redundant components, however, require additional resources, e.g., additional costs involved in terms of programming efforts, hardware requirements and time needed for design and test. During operation, redundancy causes longer processing time. In applications such as a complex chemical plant or a communications satellite system with extremely expensive parts, having additional redundancy sections may not always prove to be a practical solution.

7.1.3. Fault Diagnoses

Various methods of fault diagnoses have been developed, but most of them are concerned with digital / sequential machines- Duhamel and Rault[65]. A well known

approach to fault diagnosis for dynamic systems is to introduce some redundancies at sensors, actuators and / or the system itself in the form of a functional observer and to analyse residual signals to decide whether or not any fault exists. Chow and Wilsky[66] present a combination of methods which might be useful for the modelling and specification of the fault-tolerance properties of complex software systems. This comprises the use of advanced Petri-net constructs for the description and analysis of systems to produce regular expressions for modelling of detailed system structure. This technique permits one to investigate the error detection / correction capabilities and to formally specify the redundancy to be added to achieve fault-tolerance. Kim and Bien[69] proposed an algorithm for multiple fault diagnosis of plants (where the number of faulty units of the system is known) based on the observation that the set of faulty units can be differentiated from other sets by checking linear varieties in the measurements data space. This approach is believed to be effective for systems in which the number of units in concurrent faulty state is less than the total number of units or for systems with large time constant and the sampling period is quite long.

Also, numerous simulation models have been developed including analyses by fault simulations, but most are system-specific and offer limited system simulation. Reibman[67] introduces performability modelling, the combined analyses of reliability and performance. While some systems are always either fully operational or failed, many high reliability fault-tolerant systems degrade considerably- Siewiorek and Swarz[68]. Bell communications[70] discusses a range of degraded operating modes of local office switching systems which includes the maximum amount of downtime or complete outage, how severely and how often a system can operate in a degraded condition. Lioy[71] provides a fault collapsing method i.e., partitioning of faults into equivalence classes where only one representative fault per class must be considered in fault simulation and test generations.

7.2 Fault Simulation

The sensitivity-based algorithm developed in the previous chapter has been shown to combine performance and reliability evaluations through modelling of systems in terms of root-contours and root-sensitivity loci.

It has provided interactively fast 2-d and 3-d screen graphics as a moving picture which made it possible to analyse continuously migrating patterns of pole/zero configurations as in the case of systems with varying parameters (chapter 6). These features lend themselves conveniently to modelling systems undergoing some modes of fault which are manifested in the form of either changing and/or migrating pole/zero pattern.

Although faults are vastly diverse, nevertheless the most common of which may be classified into two main categories according to both the degree and speed of their effects on the normal system's operation:

1. Slow degradation of the system's characteristics.
2. Sudden and complete break-down of some sub-systems or components.

This classification does not necessarily rule out the possibility that the first category of fault may eventually culminate in the second if no preventative measures have been taken or it were not detected in the first place.

The second category of fault, if the system is still partially operational, would cause drastic change in either the number of the poles and zeros, their distribution in the S-plane or combination of both. Whereas the first category would cause only gradual redistribution of the same pattern of poles and zeros in the S-plane as if continuously migrating with different relative velocities in general.

Concentrating on the first category, the path of migration of every pole and zero (referred to subsequently as fault-path) will depend on the nature of the fault (or combination of faults) and its evolving dynamics. In practice, such a fault-path is difficult to predict either because of lack of knowledge of the exact nature of the fault or because of the difficulty in translating the fault dynamics into pole/zero migration paths and relative velocities. However, it is well known that gradual deterioration of system's characteristics could invariably be the symptom of slow migrations of the system's poles in the general direction towards the imaginary axis or the right-half of S-plane (please refer to the KBSC studies conducted in chapter 4). On the other hand, the direction of migrations of the system's zeros, which may concurrently occur, cannot be similarly generalised. This is because their effects on the system's characteristics are by far more subtle and indirect (through their effects on the residues) compared to those due to the poles.

In principle, therefore, it is possible to identify **a fault-region** associated with each pole and zero which encompasses all the likely migration paths even in the absence of knowledge of the precise shape and the velocity of each single fault path. For instance, figure 7.1 shows an example of the likely fault regions (indicated by shaded areas) associated with a pair of complex poles and a pair of complex zeros. For the complex poles, each fault region appears as a horizontal sector of concentric circles centred at the pole position of the unfaulty system (A & A*) and mirrors that associated with its conjugate to preserve the realisability condition. The migration path ABCDE and its conjugate A*B*C*D*E* define the upper fault-boundary, inclined at an angle "θ" to the horizontal, and the migration path AFGHI and its conjugate A*F*G*H*I* define the lower fault-boundary, inclined at an angle "$-\theta$" to the horizontal. Where the value of θ (less than $\pi/2$) depends on both the system and the nature of the likely faults or combination of faults.

Although any migration path which corresponds to specific fault-dynamics may be not necessarily easily predictable, the progression as explained earlier will be from left to right crossing each circular arc only once in the order: arc BF, arc CG, arc DH, arc EI, ...etc.

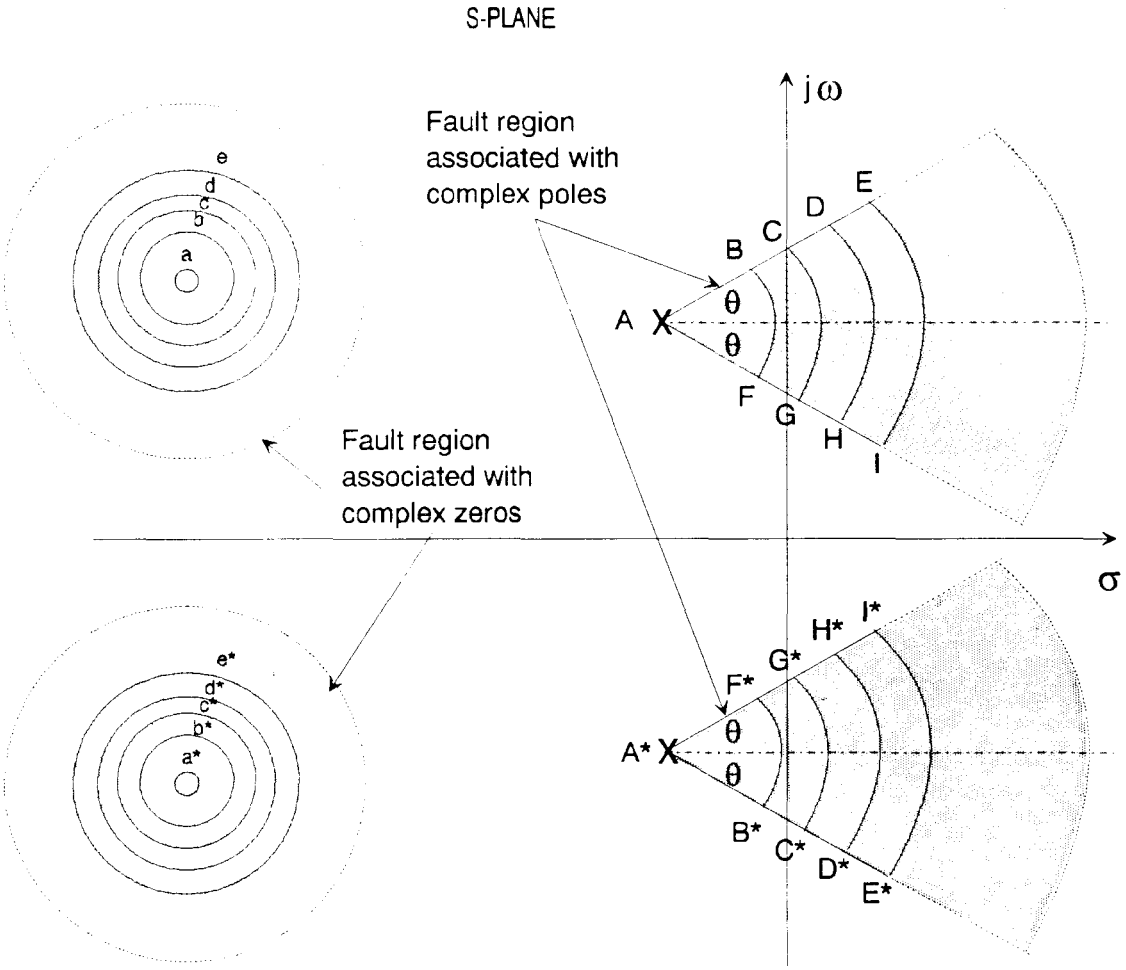


Figure 7.1 An Example Of The Likely Fault Regions Associated With Complex Poles And Complex Zeros

For the complex zeros, on the other hand, each fault region appears as concentric circles centred at the zero position of the unfaulty system (a & a*) and mirrors that associated with its conjugate for the same aforementioned realisability condition.

In contrast with the poles, the zero migration path and its progression are both likely to be unpredictable, i.e., the migration path may cross and re-cross each circular circumference at any point and in any order.

Similar to the regions of uncertainty explained in the previous chapter, the fault regions associated with real poles and real zeros will be the vertically collapsed versions of their complex counterparts so that their migration paths remain on the real axis.

As mentioned before, the exact nature of the fault or a combination of faults that may occur in a system is not likely to be predictable in general; so are the migration paths of the system's poles and zeros. However, by defining fault regions which encompass all the likely fault paths, it is possible to simulate various scenarios of faults which would allow the designer to evaluate the system's performance and its fault tolerance. These provide a valuable insight into the system's behaviour and reliability for different controller designs. These study scenarios can be made considerably efficient by selecting fault paths along the boundaries of the fault regions. These effectively subsume all the studies into two runs which correspond to the upper and lower boundaries for a specific value of " θ " (refer to figure 7.1). Since the value of " θ " is not likely to be known as a priori, it is nevertheless possible to proceed with the analysis by selecting a finite number of " θ " values which lie between 0 and $\pi/2$ radians. The corresponding "n" studies will effectively highlight the overall system's fault behaviour under all possible conditions and combinations of the first category. These studies, however, assume the degree of severity of the faults or of combinations of faults in terms of the distance of the circular arc from the unfaulty system's poles and zeros. To extend the studies to cover several degrees of severity of the faults or combinations of faults, "m" distances may be selected along the boundaries of the fault region. As a consequence, a relatively comprehensive analysis of a system operating under a fairly wide fault conditions of the first category may be conducted through

" n x m " separate studies. A stipulated range between 3 to 5 for both " n and m" should be adequate for the majority of the practical systems.

Such off-line studies, apart from providing the necessary knowledge and guidance to the designer at this preliminary stage, would, more significantly, define the design specifications for an appropriate adaptive controller which guarantees optimum system's performance and sensitivity under normal operating conditions, and in the meantime preserves the system's integrity under a wide range of plant fault conditions. The next section outlines a case study which reveals these features as well as the limitations.

7.3 Application of the Sensitivity-Based Algorithm to the Design of an Adaptive Controller For Reliable/Fault Tolerant Systems

In this section, the case study of a typical autopilot control system, which was selected in chapters 5 & 6, is again chosen here to illustrate the role of the developed sensitivity-based algorithm in:

i) the simulation of plant faults or combinations of faults of category 1,

ii) the design of an adaptive controller to deliver optimum performance together with a high degree of robustness when the system is operating under unfaulty conditions, as well as to maintain the system's integrity under a wide range of plant fault conditions.

The objective is to carry out a series of **plant fault scenarios** to evaluate the extent of the control system's reliability and tolerance beyond which the system becomes uncontrollable, for a specific controller configuration.

The unfaulty plant dynamics including the actuator have been described earlier by the deterministic model as given by equation (5.1). The plant is now assumed to undergo fault conditions in the form of slow gradual degradation of characteristics, which are manifested by random migration of the complex conjugate poles (at $-1.0 \pm j1.73$). The fault-region associated with these poles take, therefore, the shape of twin-mirrored horizontal sectors of concentric circles, as explained in figure 7.1, with an upper and a lower fault boundaries inclined at an angle " $\pm \theta$ " to the horizontal. In the subsequent study, only one value of (" θ "$=\pi/6$ radians) will be considered for the demonstration purposes as shown in the transparencies associated with figures 7.2, 7.4, 7.6 & 7.8, designated as **"plant fault path"**.

The plant faults are also assumed to be concurrently caused by a gradual increase in delay from 0 to 0.05 seconds, where the plant fault paths at the beginning and end of the delay changes are shown in figures 7.2 & 7.4 and 7.6 & 7.8, respectively.

The strategy for deciding on the controller's configuration and the adaptive control parameters follows, in the first place, a similar procedure to that outlined in section 6.4. This implies that the control parameters "ξ_A" , "ω_A" and "K_{CA}" are initially tuned to optimise the combined system's performance and sensitivity when the plant fault conditions are relatively mild (the migration of the plant's complex poles are within a distance of about 0.577 from their relevant unfaulty positions).

As the severity of the faults intensifies, the optimum strategy is relaxed and gradually replaced by a strategy that preserves the system's integrity and stability. For simplicity, the configuration (i.e., the order) of the adaptive controller has been deliberately kept unchanged. As a result, the combined two strategies have produced **decision paths** for the adaptive controller zeros as shown in the transparencies associated with figures 7.2, 7.4, 7.6 and 7.8. These have been guided by the boundary surfaces of the root contour and the root sensitivity loci produced by the sensitivity-based CAD system.

Snapshots of these diagrams for the control system are shown in figures 7.2 to 7.9 as the plant undergoes faults or combinations of faults defined earlier in general terms. With the inbuilt design strategies guided by the off-line computations and fast on-screen interactive graphics, it was finally possible for the algorithm to produce almost in real time **the operation paths** of the fault-tolerant system as shown in the transparencies associated with figures 7.2, 7.4, 7.6 and 7.8. These operation paths represent the migrations of the dominant (or nearly dominant as the severity of the plant faults increases) closed-loop complex poles of the system actively placed by the adaptive controller.

7.4 Evaluation Of The Designed Reliable/Fault Tolerant Control System For The Autopilot Case Study

Having captured the boundary root contour and root sensitivity loci corresponding to five pole migration positions at equal intervals of 0.577 (designated as a, b, c, d & e and a, f, g, h & i) along each of the plant's upper and lower boundary fault paths, it was possible to obtain from the CAD system a host of results relevant to the combined performance- reliability measures of the control system operating at the adaptively assigned closed-loop poles (see the operation paths of the fault-tolerant system in figures 7.2, 7.4, 7.6 and 7.8).

Tables 7.1 and 7.2 summarise the relevant results for zero plant's delay, while tables 7.3 and 7.4 give the corresponding results for plant's delay of 0.05 seconds.

The results clearly confirm that, for mild plant fault (in the region bounded by the circular arc bf and its conjugate b*f*), the designed adaptive control system performs optimally, satisfactorily robust, and behaves to a large extent as a second-order system.

These conclusions are further corroborated by an analysis of the time-domain system's responses to a unit-step target input (using successive system's simulations on Program CC version 4.0, reference [57]). The transient responses captured at a, a* (unfaulty plant) and b, b* (at the end of the relatively mild fault-region) are shown as the two narrowly close graphs in figure 7.10. Similarly those captured at a, a* and f, f* are shown as the close graphs in figure 7.11. Even when the plant is concurrently at the end of its stipulated delay fault, the transient-responses hardly deviated as shown by the graphs in figures 7.12 and 7.13, but keep very close to the optimum deterministic case.

For severe plant faults, however, the characteristics of the adaptive system degrades gradually with the degree of the faults severity. In addition, the system behaves more or less as a predominantly fourth-order system. The transient responses progressively widen their deviations from that which corresponds to the unfaulty plant, though still maintaining acceptable operational integrity and stability until the plant faults reach the limit of the considered fault region (bounded by the circular arc ei and its conjugate e* i*) as shown in figures 7.10 and 7.11. This is true for the plant without concurrent delay faults. For the plant with concurrent delay faults, the operational integrity of the system is still maintained albeit within a contracted fault region (bounded by the circular arc c g and its conjugate c* g*) beyond which the system becomes unstable. Within this contracted fault region, the transient responses progressively widen their deviations from that which corresponds to the unfaulty plant, but at a slightly increased rate compared with the case of the plant without concurrent delay faults.

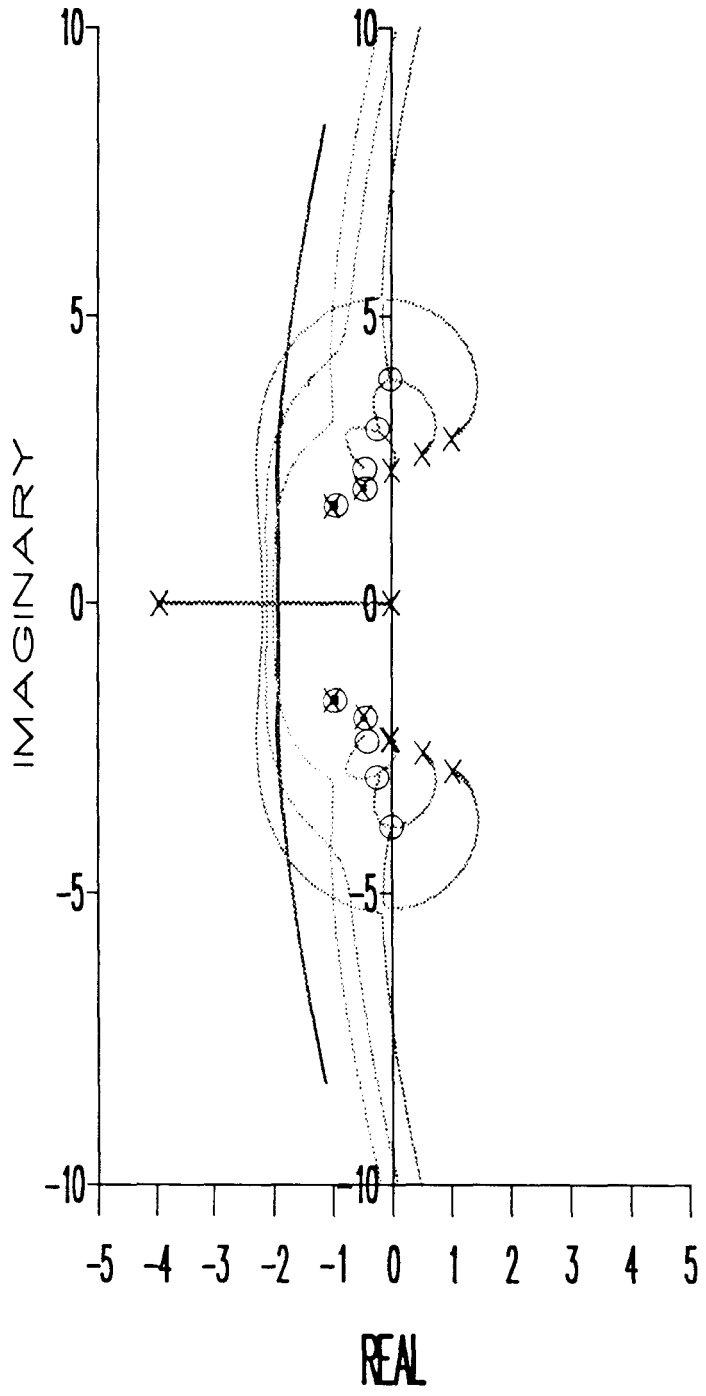


Figure 7.2 Root-Contour Diagrams Of The Adaptive Control System For A Zero-Delay Plant Undergoing Fault Of Category 1 As It Progresses Along The Upper Boundary Of The Designated Fault Region.

* The Plant Fault Path, The Adaptive Controller Decision Path And The Operational Path Of The Fault-Tolerant System, Directly Obtained By The S-B Algorithm, Are Shown On The Transparency.

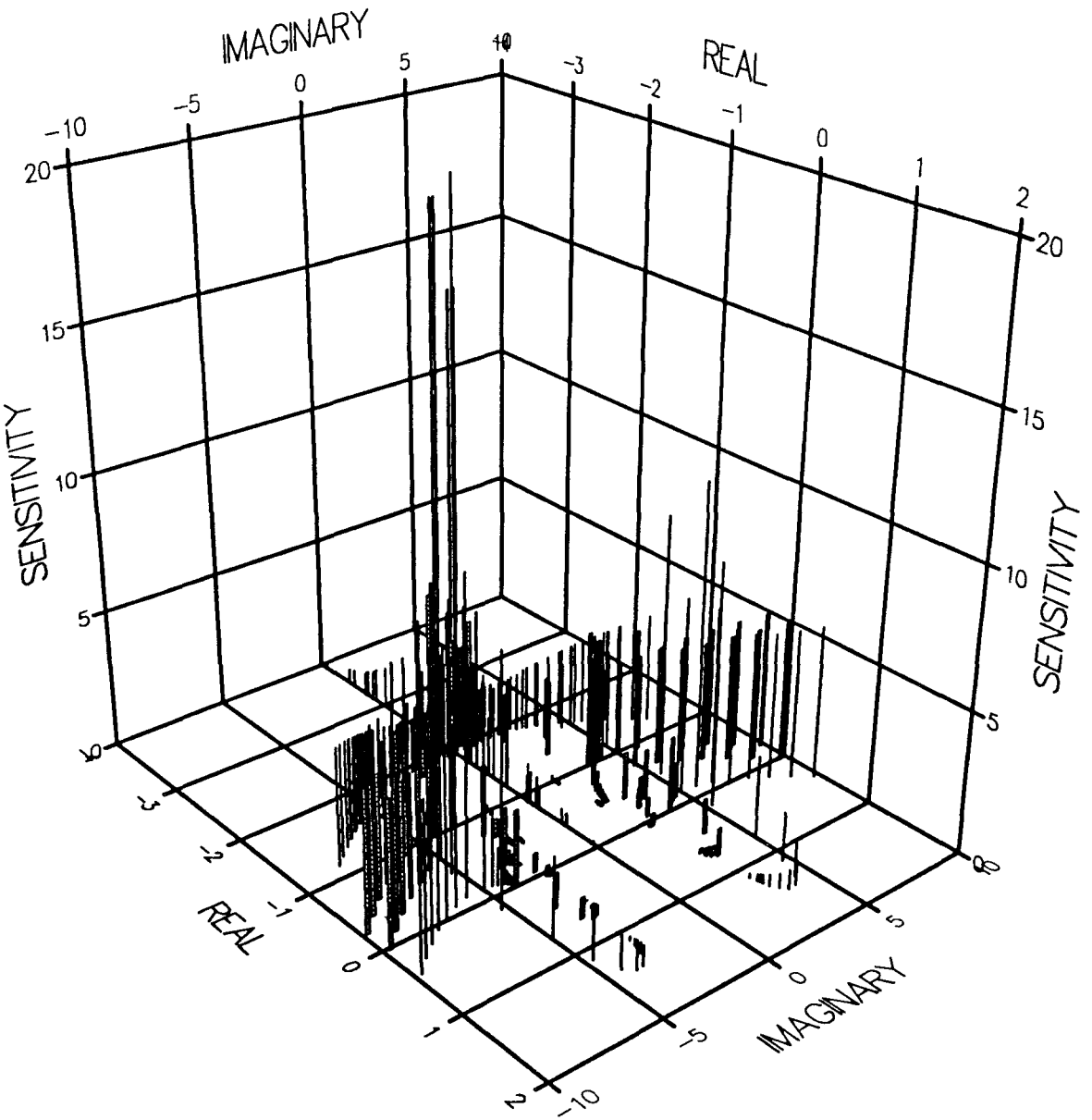


Figure 7.3 Root-Sensitivity Profiles Of The Adaptive Control System For A Zero-Delay Plant Undergoing Fault Of Category 1 As It Progresses Along The Upper Boundary Of The Designated Fault Region.

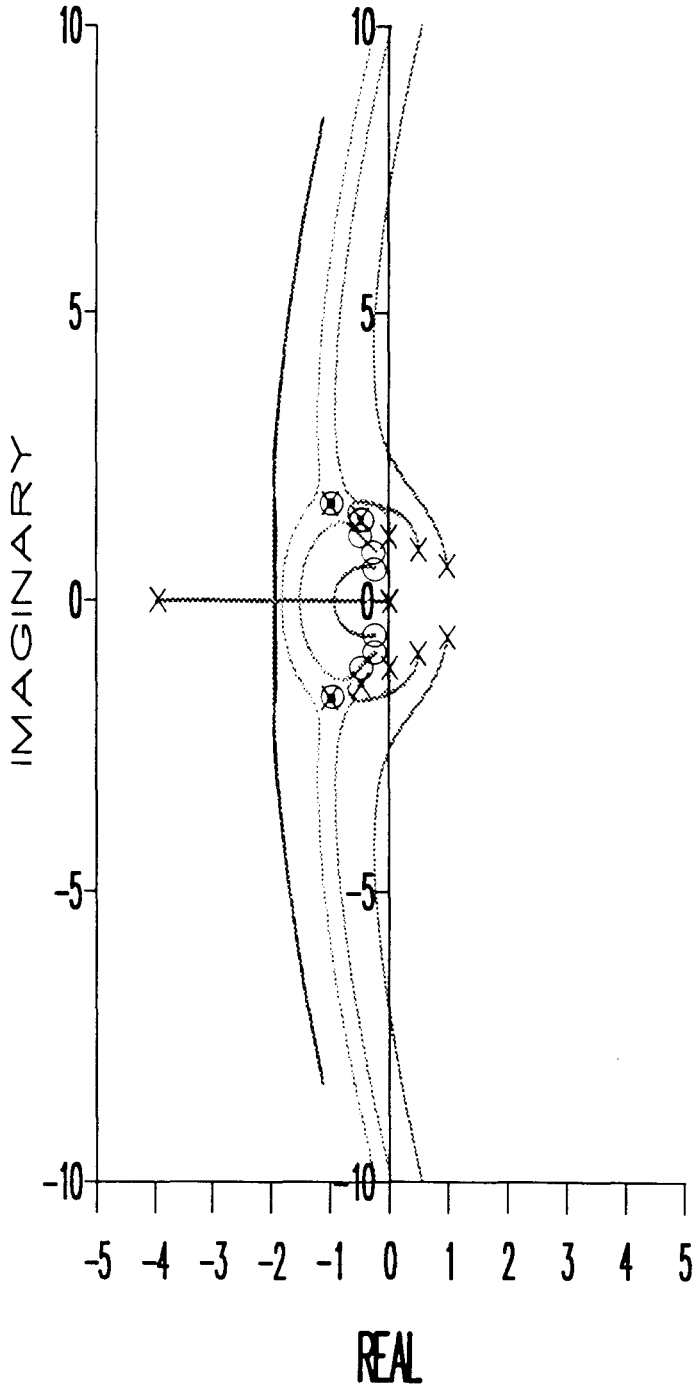


Figure 7.4 Root-Contour Diagrams Of The Adaptive Control System For A Zero-Delay Plant Undergoing Fault Of Category 1 As It Progresses Along The Lower Boundary Of The Designated Fault Region.

* The Plant Fault Path, The Adaptive Controller Decision Path And The Operational Path Of The Fault-Tolerant System, Directly Obtained By The S-B Algorithm, Are Shown On The Transparency.

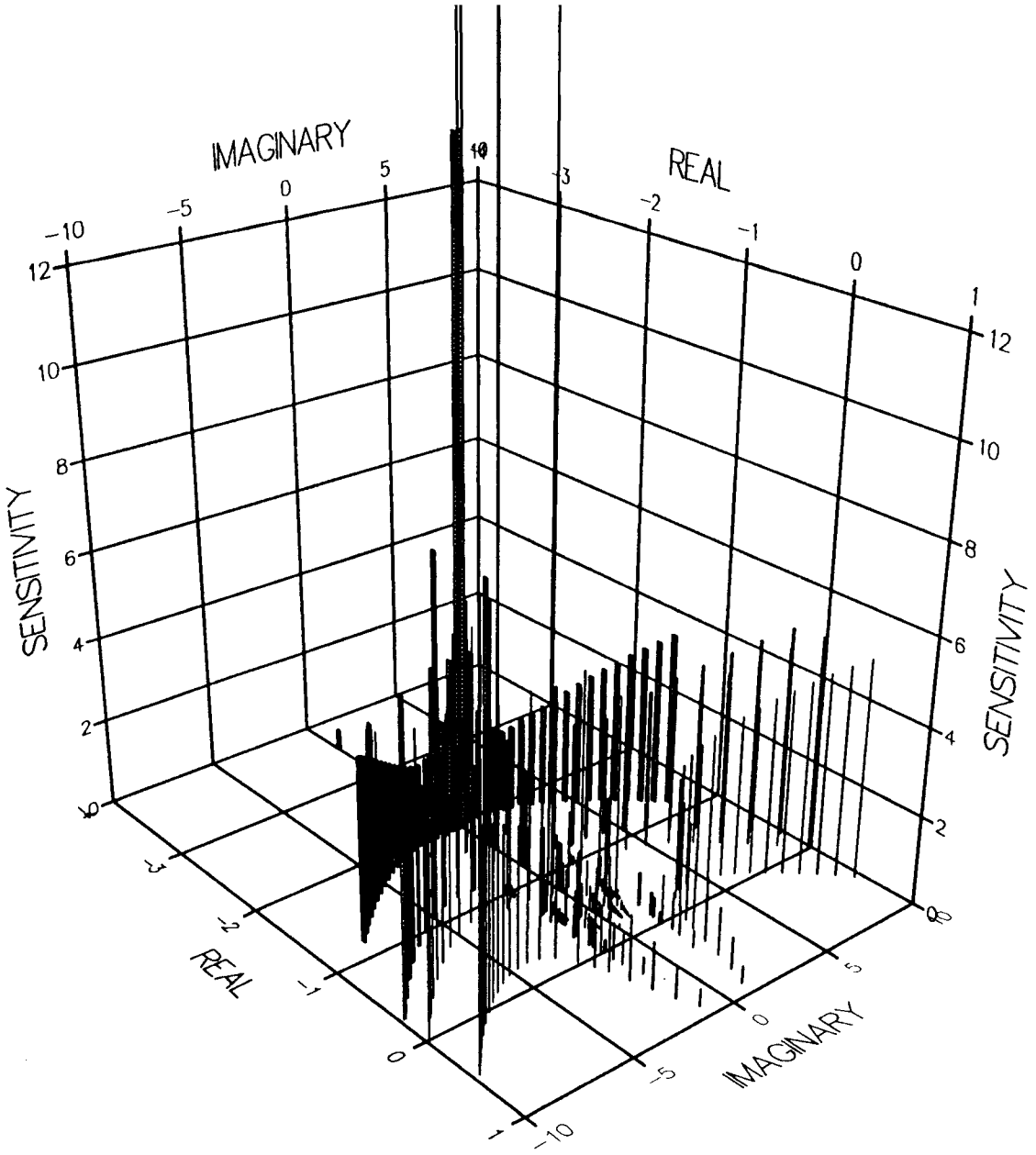


Figure 7.5 Root-Sensitivity Profiles Of The Adaptive Control System For A Zero-Delay Plant Undergoing Fault Of Category 1 As It Progresses Along The Lower Boundary Of The Designated Fault Region.

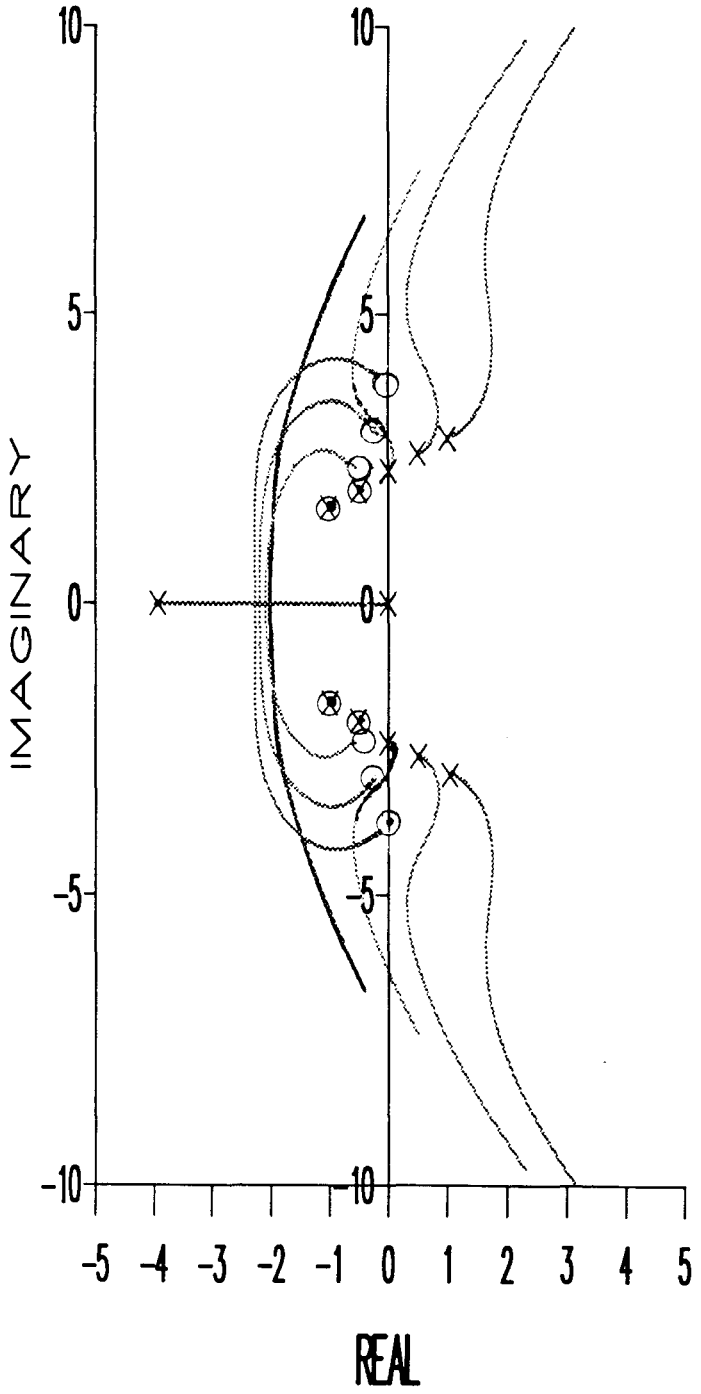


Figure 7.6 Root-Contour Diagrams Of The Adaptive Control System For A 0.05 s Delay Plant Undergoing Fault Of Category 1 As It Progresses Along The Upper Boundary Of The Designated Fault Region.

*** The Plant Fault Path, The Adaptive Controller Decision Path And The Operational Path Of The Fault-Tolerant System, Directly Obtained By The S-B Algorithm, Are Shown On The Transparency.**

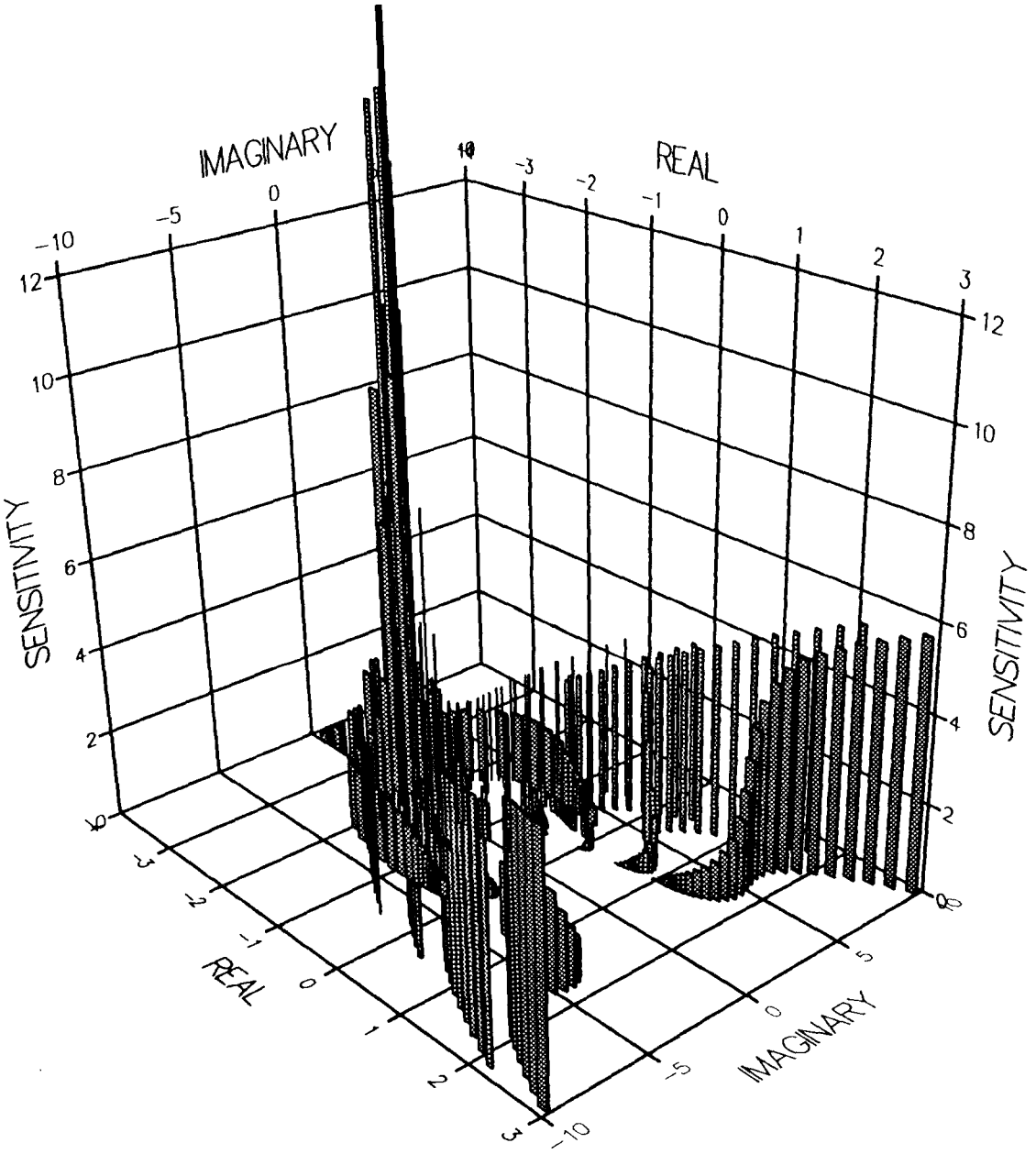


Figure 7.7 Root-Sensitivity Profiles Of The Adaptive Control System For A 0.05 s Delay Plant Undergoing Fault Of Category 1 As It Progresses Along The Upper Boundary Of The Designated Fault Region.

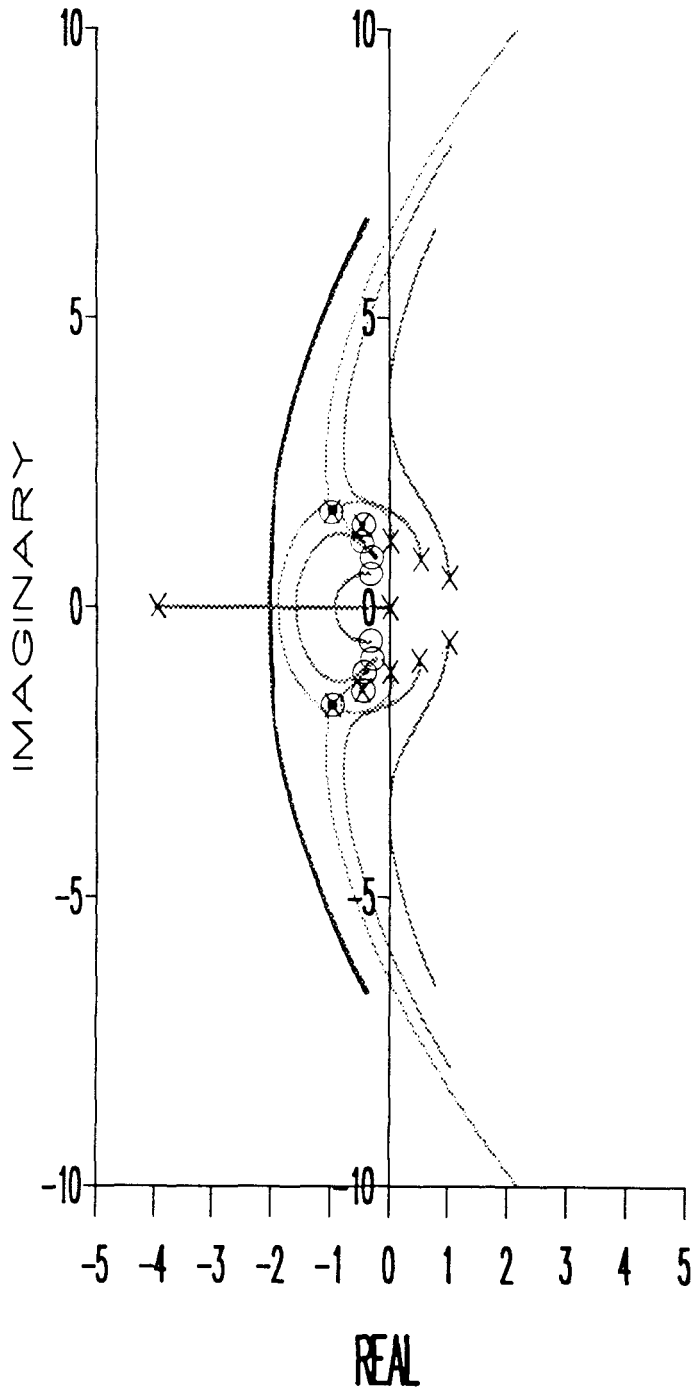


Figure 7.8 Root-Contour Diagrams Of The Adaptive Control System For A 0.05 s Delay Plant Undergoing Fault Of Category 1 As It Progresses Along The Lower Boundary Of The Designated Fault Region.

*** The Plant Fault Path, The Adaptive Controller Decision Path And The Operational Path Of The Fault-Tolerant System, Directly Obtained By The S-B Algorithm, Are Shown On The Transparency.**

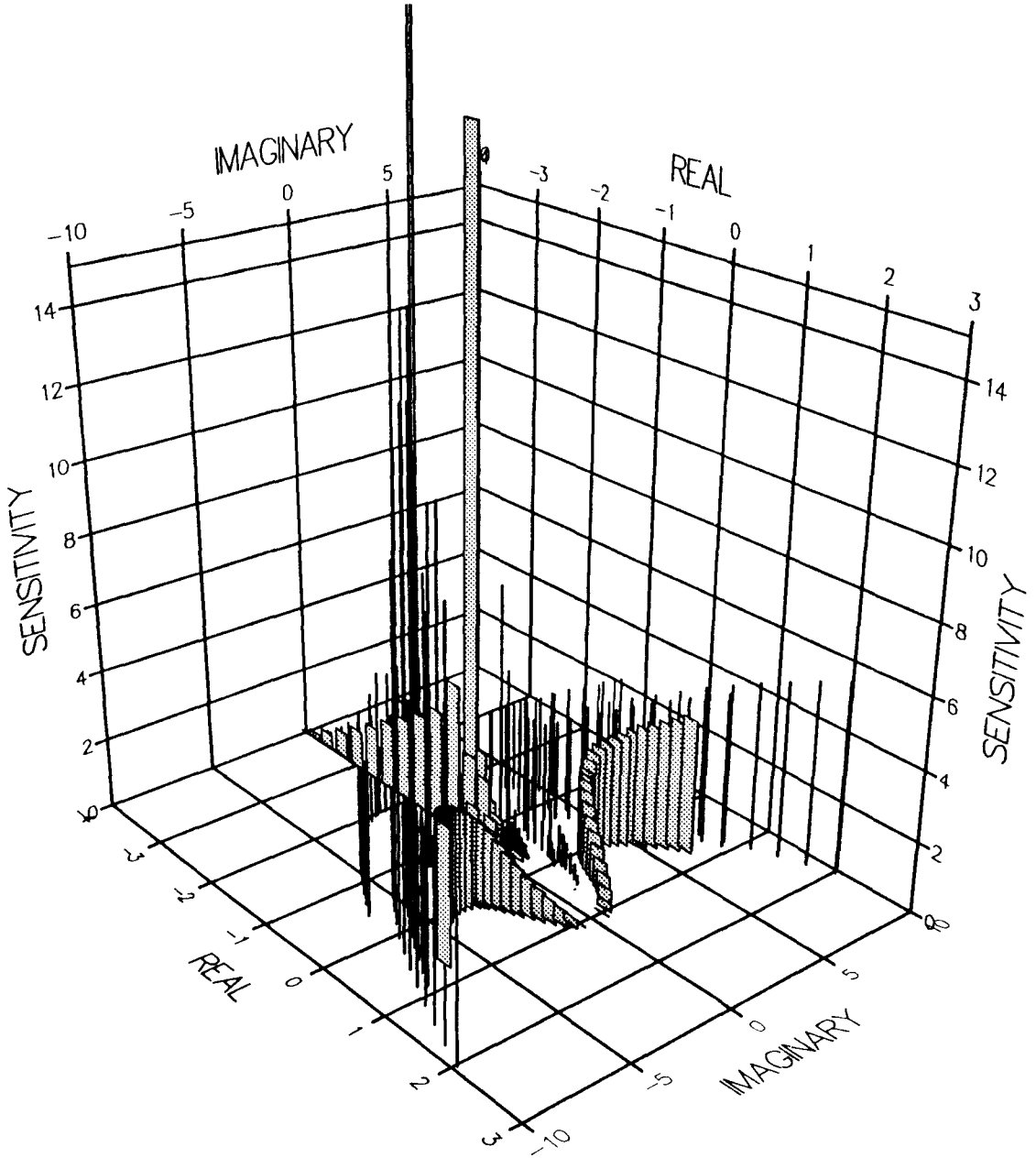


Figure 7.9 Root-Sensitivity Profiles Of The Adaptive Control System For A 0.05 s Delay Plant Undergoing Fault Of Category 1 As It Progresses Along The Lower Boundary Of The Designated Fault Region.

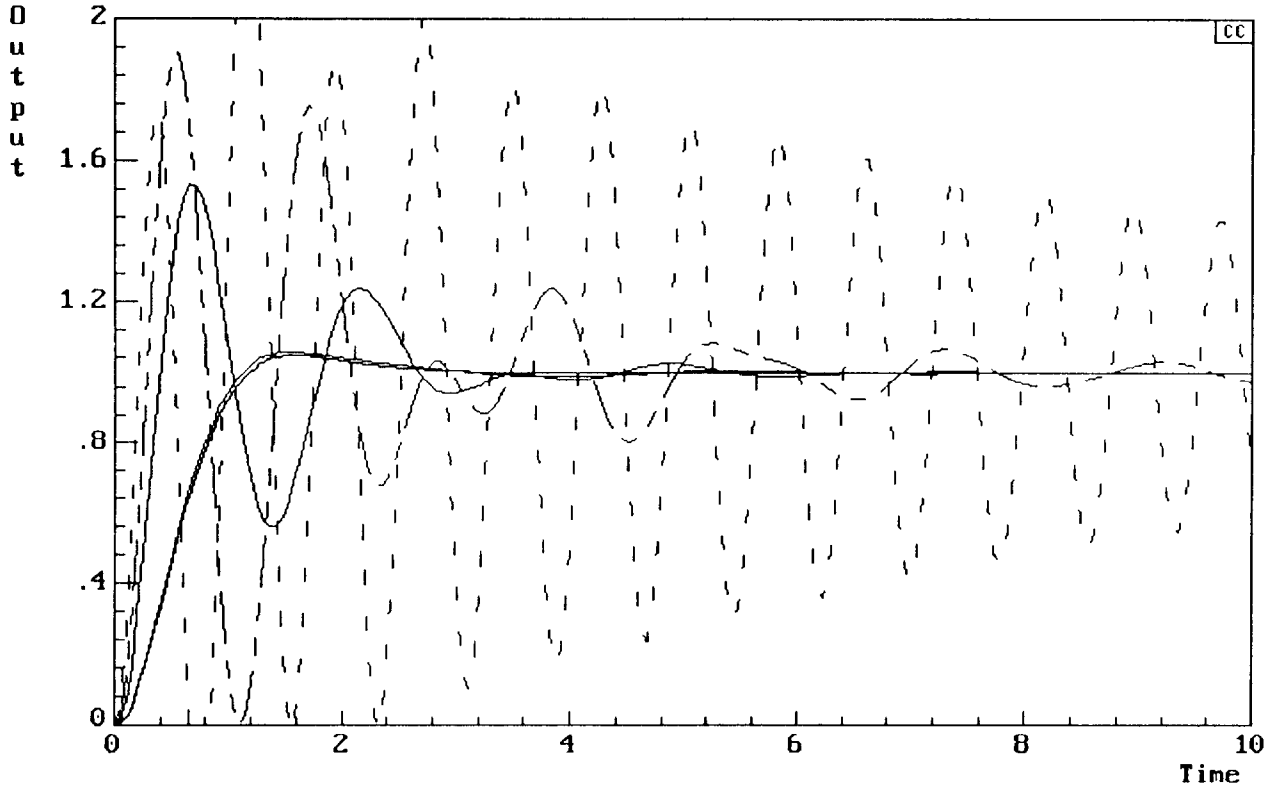


Figure 7.10 Family Of Transient Responses Of The Adaptive Control System For A Zero-Delay Plant Undergoing Fault Of Category 1 As It Progresses Along The Upper Boundary Of The Designated Fault Region.

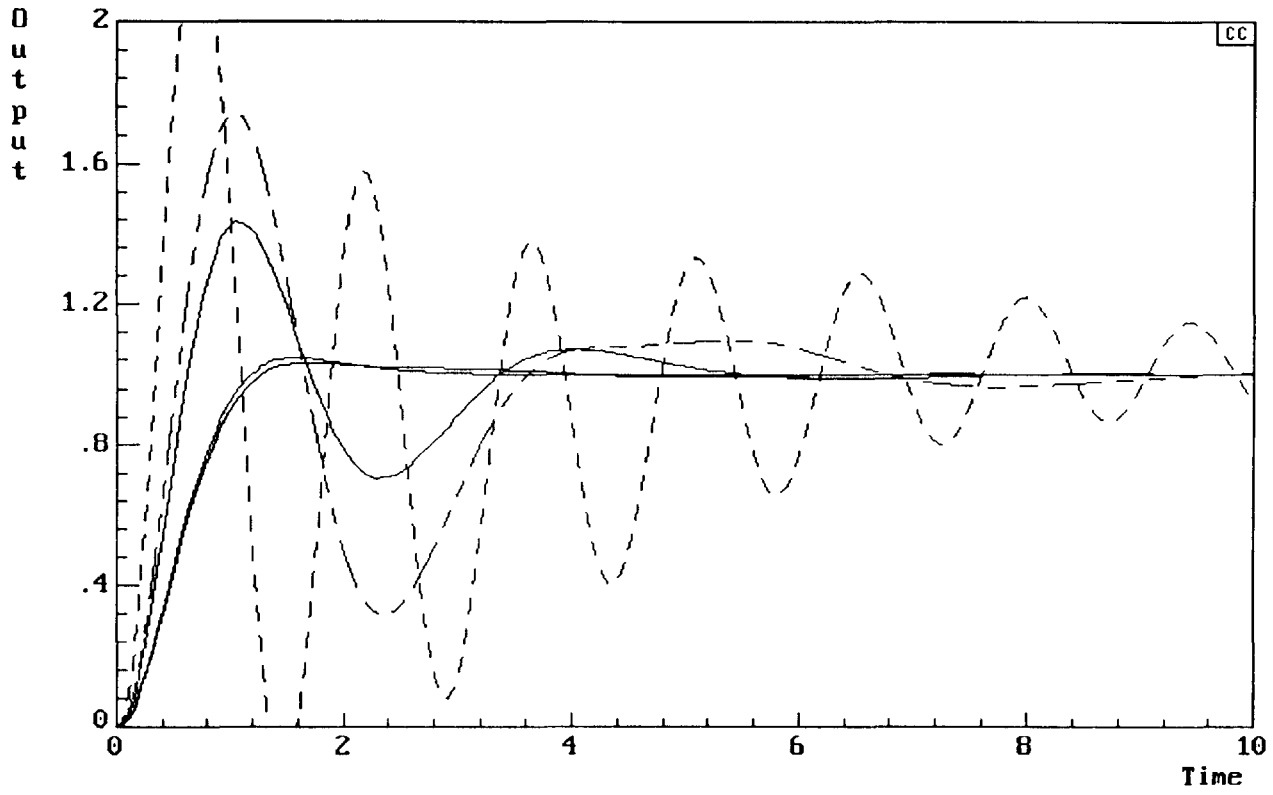


Figure 7.11 Family Of Transient Responses Of The Adaptive Control System For A Zero-Delay Plant Undergoing Fault Of Category 1 As It Progresses Along The Lower Boundary Of The Designated Fault Region.

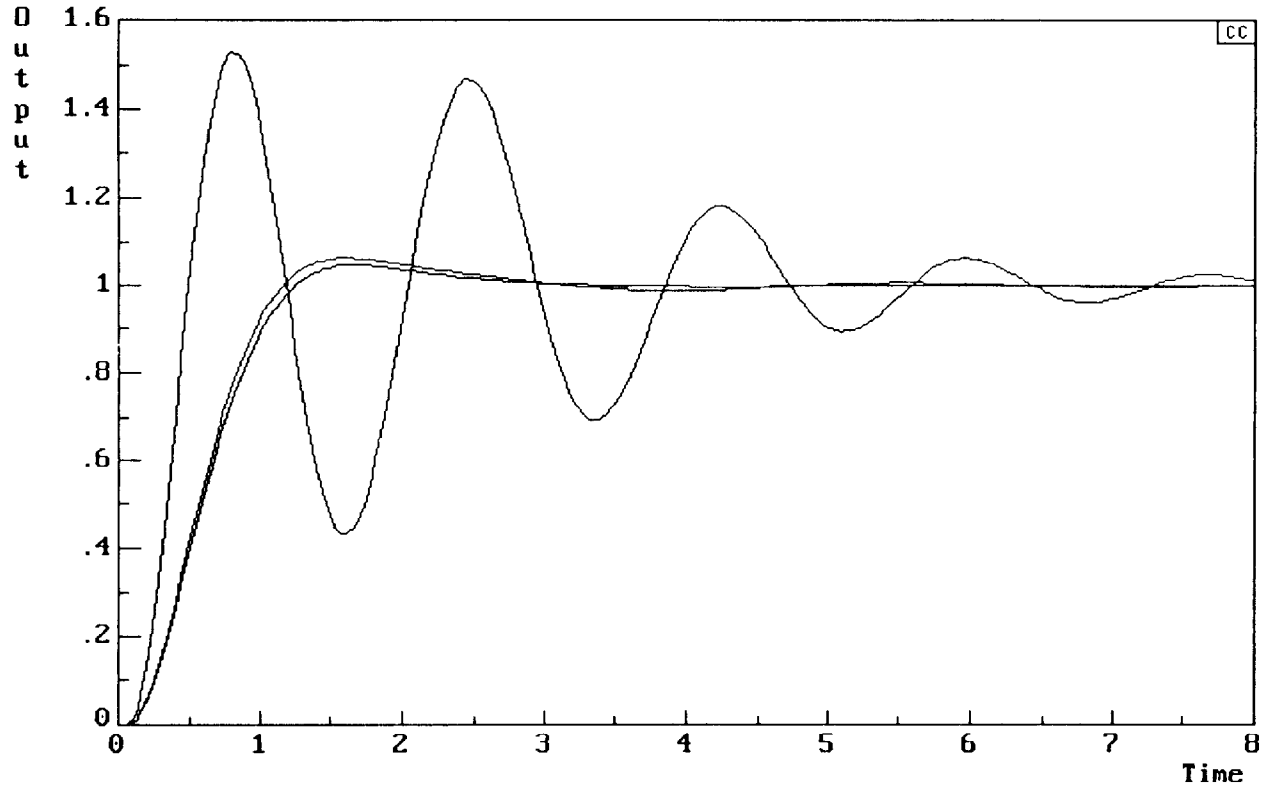


Figure 7.12 Family Of Transient Responses Of The Adaptive Control System For A 0.05 s Delay Plant Undergoing Fault Of Category 1 As It Progresses Along The Upper Boundary Of The Designated Fault Region.

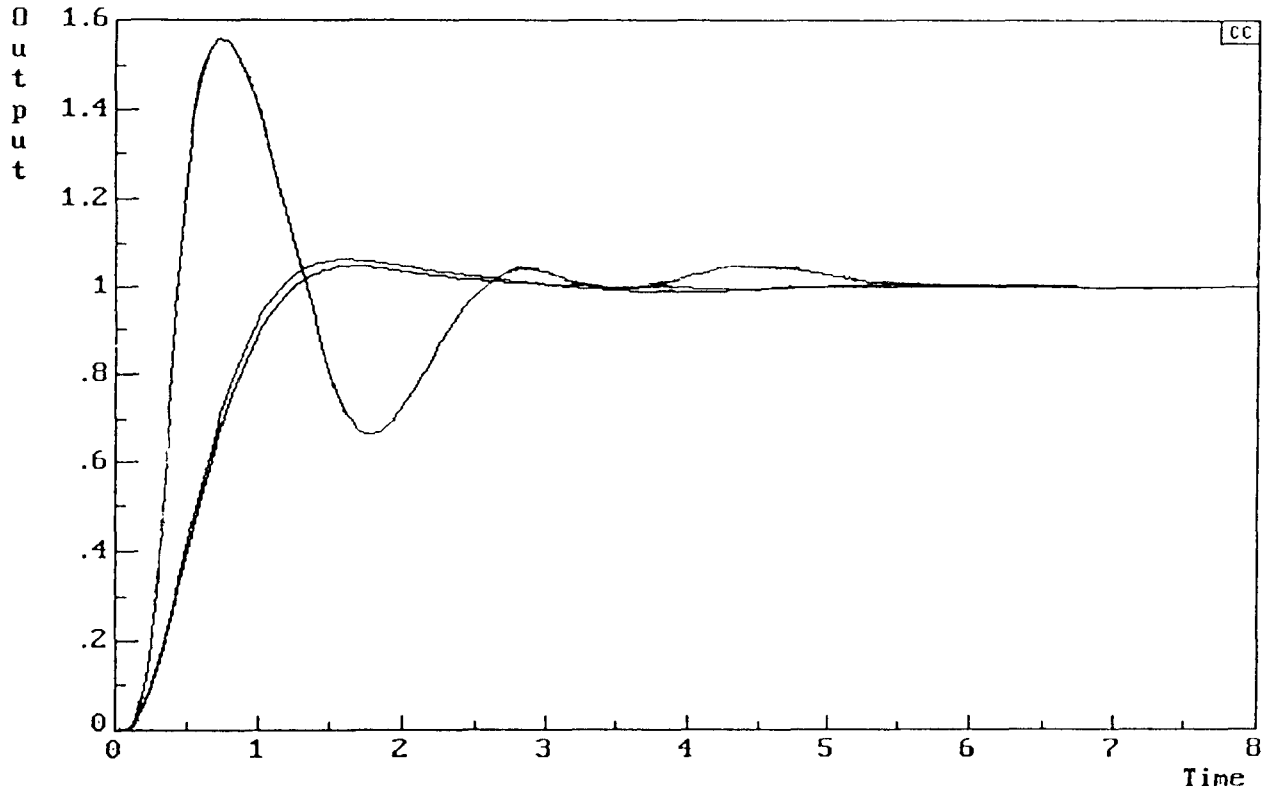


Figure 7.13 Family Of Transient Responses Of The Adaptive Control System For A 0.05 s Delay Plant Undergoing Fault Of Category 1 As It Progresses Along The Lower Boundary Of The designated Fault Region.

Table 7.1 Results Of The Adaptive Autopilot Control System's Design For A Plant Undergoing Faults Along The Upper Boundary Fault Path And With A Delay Of 0.0 Sec.(Figures Rounded To Four Decimal Places)

Plant's Complex Poles Migrations Along The Upper Boundary Fault Paths #	Sensitivity Of The C.L. Poles Emanating From Open-Loop Poles At 0 & -4.0	Bandwidth Of The C.L. Poles Emanating From Open-Loop Poles At 0 & -4.0 (rad. S^{-1})	Sensitivity Of The C.L. Poles Emanating From The Plant's Complex Poles At -1.0±j1.73	Bandwidth Of The C.L. Poles Emanating From The Plant's Complex Poles At -1.0±j1.73 (rad. S^{-1})	Loop Gain ($K_P K_{CF}$) (rad. S^{-1})	Attenuation Required Outside The Loop For Unity d.c. Gain	Velocity Error Constant K_v (rad. S^{-1})	Steady-State Error For A Unit-Ramp Input	Time-Domain Performance For A Unit-Step Target Input		
									Percent Overshoot	Settling Time (Sec.)	Rise Time (Sec.)
a, a*	1.8889	2.9474	0.1465	1.9012	324.2554	1.0000	2.0658	0.4841	4.6801	2.5352	0.7500
b, b*	1.9856	2.9809	0.0831	2.0142	339.2614	1.0002	2.0968	0.4769	5.9557	2.7671	0.7236
c, c*	3.2284	4.5457	0.5393	2.6027	1027.9521	1.0000	6.6691	0.1499	53.7230	3.4225	0.2429
d, d*	4.2511	5.6788	0.8656	3.4361	1699.0185	1.0188	13.7962	0.0724	90.7400	10.7218	0.1796
e, e*	10.9388	5.7424	7.0789	4.9376	2293.1318	1.0057	29.7435	0.0336	127.1560	37.3944	0.1232

See Figure 7.2

Table 7.2 Results Of The Adaptive Autopilot Control System's Design For A Plant Undergoing Faults Along The Lower Boundary Fault Path And With A Delay Of 0.0 Sec.(Figures Rounded To Four Decimal Places)

Plant's Complex Poles Migrations Along The Upper Boundary Fault Paths #	Sensitivity Of The C.L. Poles Emanating From Open-Loop Poles At 0 & -4.0	Bandwidth Of The C.L. Poles Emanating From Open-Loop Poles At 0 & -4.0 (rad. S^{-1})	Sensitivity Of The C.L. Poles Emanating From The Plant's Complex Poles At -1.0±j1.73	Bandwidth Of The C.L. Poles Emanating From The Plant's Complex Poles At -1.0±j1.73 (rad. S^{-1})	Loop Gain ($K_P K_{CF}$) (rad. S^{-1})	Attenuation Required Outside The Loop For Unity d.c. Gain	Velocity Error Constant K_v (rad. S^{-1})	Steady-State Error For A Unit-Ramp Input	Time-Domain Performance For A Unit-Step Target Input		
									Percent Overshoot	Settling Time (Sec.)	Rise Time (Sec.)
a, a*	1.8889	2.9474	0.1465	1.9012	324.2554	1.0000	2.0658	0.4841	4.6801	2.5352	0.7500
f, f*	1.9121	2.8608	0.0591	1.4655	305.0256	1.0300	1.8695	0.5349	3.3750	3.6549	0.7817
g, g*	2.9457	2.7949	0.7631	1.5473	482.2769	1.0224	3.4742	0.2878	44.8250	5.3028	0.4014
h, h*	0.7966	1.1942	3.0578	2.7026	532.4372	1.0038	2.6264	0.3807	74.2550	8.8943	0.3380
i, i*	0.3059	0.7234	3.0409	4.3360	1081.0190	1.0150	1.8539	0.5394	124.0430	17.1127	0.1901

See Figure 7.4

Table 7.3 Results Of The Adaptive Autopilot Control System's Design For A Plant Undergoing Faults Along The Upper Boundary Fault Path And With A Delay Of 0.05 Sec.(Figures Rounded To Four Decimal Places)

Plant's Complex Poles Migrations Along The Upper Boundary Fault Paths #	Sensitivity Of The C.L. Poles Emanating From Open-Loop Poles At 0 & -4.0	Bandwidth Of The C.L. Poles Emanating From Open-Loop Poles At 0 & -4.0 (rad. S^{-1})	Sensitivity Of The C.L. Poles Emanating From The Plant's Complex Poles At -1.0±j1.73	Bandwidth Of The C.L. Poles Emanating From The Plant's Complex Poles At -1.0±j1.73 (rad. S^{-1})	Loop Gain ($K_P K_{CF}$) (rad. S^{-1})	Attenuation Required Outside The Loop For Unity d.c. Gain	Velocity Error Constant K_v (rad. S^{-1})	Steady-State Error For A Unit-Ramp Input	Time-Domain Performance For A Unit-Step Target Input		
									Percent Overshoot	Settling Time (Sec.)	Rise Time (Sec.)
a, a*	1.8785	2.9266	0.1699	1.8948	354.7829	0.8908	2.1594	0.4631	4.6810	2.6401	0.7605
b, b*	1.9843	2.9809	0.1002	2.0071	379.8093	0.8828	2.3474	0.4260	6.2770	2.8521	0.7183
c, c*	1.7755	2.8035	3.2647	3.6808	850.7390	0.9056	5.5194	0.1812	52.2340	8.0493	0.2746
d, d*	Unstable										
e, e*	Unstable										

See Figure 7.6

Table 7.4 Results Of The Adaptive Autopilot Control System's Design For A Plant Undergoing Faults Along The Lower Boundary Fault Path And With A Delay Of 0.05 Sec.(Figures Rounded To Four Decimal Places)

Plant's Complex Poles Migrations Along The Upper Boundary Fault Paths #	Sensitivity Of The C.L. Poles Emanating From Open-Loop Poles At 0 & -4.0	Bandwidth Of The C.L. Poles Emanating From Open-Loop Poles At 0 & -4.0 (rad. S^{-1})	Sensitivity Of The C.L. Poles Emanating From The Plant's Complex Poles At $-1.0\pm j1.73$	Bandwidth Of The C.L. Poles Emanating From The Plant's Complex Poles At $-1.0\pm j1.73$ (rad. S^{-1})	Loop Gain ($K_P K_{CF}$) (rad. S^{-1})	Attenuation Required Outside The Loop For Unity d.c. Gain	Velocity Error Constant K_v (rad. S^{-1})	Steady-State Error For A Unit-Ramp Input	Time-Domain Performance For A Unit-Step Target Input		
									Percent Overshoot	Settling Time (Sec.)	Rise Time (Sec.)
a, a*	1.8785	2.9266	0.1699	1.8948	354.7829	0.8908	2.1594	0.4631	4.6810	2.6401	0.7605
f, f*	1.9098	2.8391	0.0683	1.4618	333.4838	0.9073	2.0439	0.4892	3.4040	3.5704	0.8239
g, g*	2.6956	3.5228	0.3582	1.4197	708.4051	0.9249	5.1031	0.1959	55.1060	5.1761	0.3169
h, h*	Unstable										
i, i*	Unstable										

See Figure 7.8

7.5 Conclusions

The ever increasing demand for new and more sophisticated products and services at an economical price has forced the pace of the development of yet more complex and invariably costly plants. This evolution process has been recently tempered by the impact on health, safety and on the environment at large; posing a challenge to the designers to bring into being intelligent systems featuring higher reliability, adaptability, efficiency and fault tolerance than ever before.

To address these issues, a generalised procedure for the simulation of a fairly common class of fault that causes slow degradation of performance in real plants has been developed. Fault regions of plant pole/zero migrations have been identified.

Efficient analysis along adaptable fault boundaries has been proposed to cater for diverse fault or combinations of fault scenarios, falling within this class. The interactive graphic facilities together with the degree of the computational automation and decision making embedded in the sensitivity- based algorithm design have been instrumental in implementing the procedure.

It was recognised, however, that plant fault simulation and analysis represent only a stepping stone towards achieving the ultimate objectives, namely:

> "design and evaluation of an adaptive controller to deliver optimum performance together with a high degree of robustness when the system is operating under unfaulty conditions, as well as to maintain the system's integrity under a wide range of plant fault conditions."

To this end, a case study of a typical autopilot control system was considered to illustrate an integrated approach to the problem. The results obtained with the help of the multi-role of the sensitivity-based algorithm as a CAD system, and which have been

corroborated by time-domain simulations using Program CC version 4.0, have clearly demonstrated that the aforementioned goals have been achieved. Namely that a realisable adaptive controller can be found which guarantees a satisfactory robust system performance under relatively mild plant faults, yet preserves the system's integrity for relatively sever plant faults. Even with a deliberately restricted controller configuration (order), it has been shown that it was still possible to maintain the system's reliability and fault tolerance over a relatively wide range of faults or combinations of concurrent faults.

The work has highlighted the ability of the developed sensitivity-based algorithm to produce, almost in real-time, the optimum **decision path** for the adaptive controller zeros and the resulting **operation path** of the fault-tolerant system's closed-loop poles for a particular **plant fault path**. This information constitutes some of the essential ingredients for a possible future adaptation of the algorithm to operate as on-line intelligent controller to diverse real-life applications.

Chapter 8

Conclusions

8.1 Introduction

The elevation of health, safety and environmental perspectives in the recent years have focused our attention to the need for a fundamental review of our design criteria and techniques. It is no longer sufficient to aim for optimum performance, robustness and cost effectiveness. Reliability, adaptability and fault tolerance have become at least equal if not higher key objectives particularly for new sophisticated applications. To achieve this, it was necessary to advance some aspects of plant modelling techniques and man-machine interactions, on one hand, and to establish an integrated method for the design with an inbuilt performance evaluation and quality assurance measures, on the other.

This thesis has therefore been organised to reflect the above theme from the theoretical development of a novel application-independent sensitivity-based algorithm which incorporates modelling of plants with uncertainty and/or undergoing some common modes of faults, automatic design decisions- in terms of both performance and sensitivity-, analysis, and design assessments followed by software implementation through to generation of composite sensitivity-performance knowledge-base graphical classification of feedback systems, and finally demonstrating its comprehensive CAD facilities through its application to an autopilot control system design which guarantees robust performance for stochastic plants and/or for a plant undergoing a relatively mild

fault modes, and which can be self-adapted to maintain the system's integrity when the plant undergoes a relatively sever fault modes.

8.2 Summary Of The Work Done

In the introduction chapter, a brief review of the most recent literature, on the classification of sensitivity and its role as a design criterion for systems with plant parameter variations, has been outlined. In particular, the root sensitivity (or eigenvalue sensitivity) has received considerable attention although it is a less direct measure of system performance. This is mainly because of the widespread use of pole-placement or eigenvalue assignment design-based techniques which give the designer the ability to specify system stability and general characteristics of the time-domain response. The investigators have unanimously reported that the incorporation of a sensitivity criterion has enhanced the characteristics of their particular designs through modal insensitivity. However, to the best of our knowledge, no attempt so far has been made to incorporate the sensitivity within a generalised design/analysis framework suitable for diverse control system applications in real-life environment. In this context, the sources and the consequences of plant uncertainty have been summarised and the relevant models have been briefly discussed.

Chapter 2 has been devoted primarily to the theoretical development of a novel sensitivity-based root contour algorithm. Central to this work is the derivation of explicit relationships which link the residue at the closed-loop poles of a system to their respective sensitivities due to small variations in any single or combinations of loop gain, open-loop poles, open-loop zeros and plant delay.

The sensitivity formulations due to any single or composite parameter variations were then used to predict the new locations of the systems closed-loop poles. Therefore, by

updating the values of the residues, both the root-contour and the root sensitivity loci were made possible to be computed **in parallel**.

The main advantage of this new algorithm is that it contains an inbuilt efficient and adaptable structure which:

1. lends itself to a relatively easy software implementation for fast computations and for highly automated or interactive composite screen graphical displays of 2-d and 3-d performance-sensitivity profiles almost in real time,
2. allows the provision of a host of default criteria such as the step size for a predetermined resolution, the number of steps (or window size) and the best angle of viewing perspective of the 3-d graphics and on-screen composite displays with the option of over-riding any or all by the user,
3. allows a high degree of automated procedures and direct access to a fairly comprehensive system analysis,
4. allows easy incorporation of specification-loci and design strategies for both automatic and interactive decisions (as demonstrated by the applications in the last three chapters).

The detailed implementation of the algorithm using the high-level software language Turbo-C, is outlined in chapter 3. This has been designed as a user friendly CAD tool, incorporating highly adaptable graphic facilities and on-screen windows prompted by a host of command menus, to run on a personal computer for both cost-effectiveness and portability. The complete program listing is given in Appendix A.

Apart from testing at the local level for program debugging, a comprehensive range of plant and/or system configurations has been examined primarily for the purpose of evaluation of the various aspects of the algorithm and their implementations. Some of these examples, whose root-loci are analytically predictable, were used as bench-marks

for validation of the computer implementation. Examining the composite root-contour root-sensitivity loci, however, it became evident that such graphics reveal a relatively comprehensive insight into system's behaviour which could be useful as a valuable design tool. Chapter 4 shows how these graphics were used as a basis for generating a composite sensitivity-performance knowledge-base classification of single-loop feedback systems, having deterministic plant configurations of increasing order, different open-loop pole/zero patterns and incorporating different delays.

Appendix B presents a part of this graphic knowledge-base and its general organisation which may be produced off-line and stored in the computer memory as an integrated part of the developed CAD system.

Chapter 5 investigates the deployment of the sensitivity-based algorithm as a CAD tool for controlling deterministic plants. A case study of a real-life plant was conducted with the objective to demonstrate certain aspects of the CAD facility of the algorithm for optimum controller design purposes and to highlight its significance as both an efficient and an effective tool for the designer/analyst. The case study refers to the design of a typical autopilot control system when the aircraft dynamics including its actuator are approximated by a deterministic model.

Selecting a controller pole/zero configuration from the KBSC and incorporating it with the plant in a closed-loop system, the sensitivity-based CAD system produced a screen moving picture (in the form of rapidly modified graphical displays) in the presence of performance specification loci. This allowed viewing in real-time the effects on both the performance and robustness as the controller pole/zero pattern varied in the S-plane, which resulted in optimum placing of the controller pole/zero pattern. The results captured from the sensitivity-based CAD system showed that all the required specifications have been consistently satisfied even after allowing for relatively wide variations of both plant gain ($\pm$ 5%) and delay (in the range of 0.0 to 0.2 sec.).

Concurrently and equally important, the designed controller has also narrowed considerably the variations of all the indices optimally; resulting in a fairly robust control system.

To corroborate these results, the designed control system has been simulated (using Program CC version 4.0, see reference [57]), and a series of d.c. gain normalised transient responses were obtained for a unit-step pitch target angle. The results extracted from these simulations were then subjected to extensive analysis. The analysis provided a further support to the quality assurance of the control system behaviour, designed using the developed sensitivity-based CAD system, where the mean deviation and the mean variance do not exceed 0.0163 and 0.00091 respectively (which occurred at a loop gain of 5% above its optimum value and at a plant delay of 0.2 seconds).

Since variations of gain and delay represent only limited aspects of uncertainties encountered in plants operating in the real environment, the study was, therefore, extended to encompass the optimum robust design of control systems incorporating plants subjected to all types of uncertainty, These are reflected not only by variations of gain and delay, but also by random migrations of their poles and zeros within fuzzy zones in the S-plane. The ranges of such uncertainties are limited in practice, However, and can invariably be determined. It follows, therefore, that the size of these fuzzy zones is limited.

Using these interpretations, it was possible to adapt the sensitivity-based algorithm to model plant uncertainties, as outlined in section 6.2, following a standard procedure. The objective of the rest of this chapter was, then, targeted to the design strategy that should be adopted and to the role of the S-B algorithm in providing such strategy and the method of evaluation leading to either a fixed- configuration controller or an adaptive controller design.

Although the robust control system design procedure for plants with uncertainties clearly present a considerably more complex problem compared with that of deterministic counterparts, the suitability of the CAD system to this task has been proved in view of the inbuilt adaptability and diverse facilities.

The comprehensive studies carried out on the autopilot control system design have established the design and evaluation procedures for both fixed-configuration controller and adaptive controller which optimise not only the time-domain performance indices, but also the robustness qualities of the control system when operating in a real environment. The real-time computations of the boundary surfaces of the root contour and the root sensitivity loci, accompanied with fast on-screen interactive graphics and easy access to evaluations procedures, represent key developments to assist a host of entwined design decisions.

It may be of interest to point out that the analyses have shown that, even with a deliberately gain constraint, the adaptive control system can achieve superior robust qualities for plants with uncertainties compared with the fixed-configuration optimum control system for deterministic plants.

Imparting more than one set of design strategies to the sensitivity-based algorithm was recognised as an essential requirement to assist build systems that can reliably sustain a certain performance level throughout their expected lives even in the event of fault or combinations of fault occurrence. To address this problem, a generalised procedure for the simulation of a fairly common class of fault that causes slow degradation of performance in real plants has been developed in chapter 7. Fault regions of plant pole/zero migrations have been identified.

Efficient analysis along adaptable fault boundaries has been proposed to cater for diverse fault or combinations of fault scenarios, falling within this class. The interactive graphic facilities together with the degree of the computational automation and decision making embedded in the sensitivity- based algorithm design have been instrumental in implementing the procedure.

It was recognised, however, that plant fault simulation and analysis represent only a stepping stone towards achieving the ultimate objectives, namely:

> "design and evaluation of an adaptive controller to deliver optimum performance together with a high degree of robustness when the system is operating under unfaulty conditions, as well as to maintain the system's integrity under a wide range of plant fault conditions."

To this end, a case study of a typical autopilot control system was considered to illustrate an integrated approach to the problem. The results obtained with the help of the multi-role of the sensitivity-based algorithm as a CAD system, and which have been corroborated by time-domain simulations using Program CC version 4.0, have clearly demonstrated that the aforementioned goals have been achieved. Namely that a realisable adaptive controller can be found which guarantees a satisfactory robust system performance under relatively mild plant faults, yet preserves the system's integrity for relatively sever plant faults. Even with a deliberately restricted controller configuration (order), it has been shown that it was still possible to maintain the system's reliability and fault tolerance over a relatively wide range of faults or combinations of concurrent faults.

The work has highlighted the ability of the developed sensitivity-based algorithm to produce, almost in real-time, the optimum **decision path** for the adaptive controller zeros and the resulting **operation path** of the fault-tolerant system's closed-loop poles for a particular **plant fault path**. This information constitutes some of the essential

ingredients for a possible future adaptation of the algorithm to operate as on-line intelligent controller to diverse real-life applications and meet health, safety and environmental standards.

8.3 Proposals For Future Research

8.3.1 Software Enhancement For On-Line Intelligent Controller Applications

The sensitivity-based algorithm has been designed primarily to operate in off-line mode as an accurate and efficient CAD system incorporating simulation tools of plants with uncertainty and plants undergoing a class of fault of gradual deterioration of characteristics and delays. It produces continuous flow of interactive composite 2-d and 3-d complex-frequency domain graphical displays which translate combined changes in control systems' time-domain performance and sensitivities, design specifications requirements, and criteria of optimisations. Using the existing facilities, it was possible to generate a composite sensitivity-performance knowledge-base for system configurations of increasing order. Accessing this knowledge-base should assist in the selection of the most appropriate controller's configuration prior to optimum placing of its roots.

As it stands, however, the algorithm has been implemented using Turbo-C language to run on a personal computer under MS-DOS operating system. Flexibility in terms of automation, integration, extendibility, re-usability and portability can be considerably enhanced through the use of object oriented techniques. As a consequence, future advances in Meta-architecture for human-computer interactions may be easily incorporated.

In addition, the algorithm could be made to run in on-line mode either as criteria-driven or as criteria-updating. In criteria-driven mode, the algorithm will essentially perform the task of an adaptive optimum controller for plants with wide range of parameter variations operating in a noisy environment. In criteria-updating mode, on the other hand, the adaptive algorithm would vary its controlling strategy to plants undergoing some types of fault condition by relaxing the overall system's optimum requirements, provided that appropriate fault-diagnoses were developed and incorporated. Future research may be, therefore, worthwhile directed towards this area.

8.3.2 Provision Of Multi-Perspective For Both The CAD System And The Graphic Knowledge-Base

Significantly, the algorithm development, outlined in this thesis, has been shown to rely on up-dating the estimates of the system's S-domain **residues**. Since S-domain $\rightleftharpoons$ Z-domain transformations, on one hand, and Z-domain $\rightleftharpoons$ time-domain transformations, on the other, are known to rely also on the system's residue, the algorithm may be easily modified to incorporate all these domains together with the frequency-domain (which can be directly deduced from the S-domain). In this way, a much wider range of techniques for systems' modelling, simulation, analysis and design can be used to advantage by highlighting different perspectives of the characteristics, and through provision of diverse comparative studies and corroborating results. These advantages can be similarly extended to the graphic knowledge-base classification of feedback systems.

8.3.3 A Cad System For Multi-Input Multi-Output Control Systems

The development of the sensitivity-based algorithm and the graphic knowledge-base presented in this thesis have been confined to single-input systems. They have been, however, designed with a flexible structure embodying parallel processing at low level to accommodate future additions and modifications.

It is, therefore, proposed to extend the CAD system to cater for multi-input multi-output control systems using hierarchical parallel processing techniques with interactive modular structure which shares common facilities already provided.

8.3.4 Adaptive Operational Criteria

Design criteria, by their very nature, seek to modify the parameters of a system or sub-systems so that an optimum compromise between a set of likely conflicting performance indices is reached.

On the basis of this definition, they are inherently multi-decision and are generally application-dependant. With the advent of reliable fault-tolerant systems, the criteria are unlikely to be known precisely as a priori, nor should they remain unchanged throughout their expected lives.

The work produced in chapters 6 and 7, however, may be used as a basis for developing an algorithm for automatic adaptation of a basic criterion or a combinations of several criteria to be fed by the user with continuos reference to the stored knowledge-base. It is also likely that an automatic system identification would be required.

8.3.5 Automatic Fault Identification

In referring to section 7.2, it has been shown that a common class of plant fault may be modelled in terms of continuous migration of the plant's poles and zeros. The fault-path of each pole and zero and the associated relative speed of migration are both unlikely to be predictable. In practice, however, such information could be valuable in tracing the fault back to its origin in a specific component or components.

The development of an automatic fault identification system is therefore proposed. The research in this area would benefit from the work done in chapter 4 on the composite knowledge-base classification of systems and that in chapter 7 on fault simulation and design of fault-tolerant systems.

Appendix A

Program Listing Of The

S-B CAD System

```
/*****************************************************************************
* Author          :  N Golesorkhi                                           *
* Title           :  Sensitivity-Based Algorithm                            *
* Date Written    :  11.02.89                                               *
* Revised         :  06.02.92                                               *
* ------------------------------------------------                          *
* Description     :  This Program Calculates The Systems Closed-Loop        *
*                    Poles And The Associated Sensitivity Profiles And The Amounts *
*                    Of changes To The C.L. Poles As A Result Of Any Or     *
*                    Simultaneous Variations Of Loop Gain, Open-Loop Poles, O.L. *
*                    Zeros And Time Delay Automatically Selects A Window Of *
*                    Calculation Corresponding To The Pole/Zero Pattern, Checks *
*                    The Error Angle Criteria For The Required Accuracy, Handles *
*                    High-Order O.L. Poles & Zeros And Breakaway Points. Stores *
*                    Data On a:DD.D" And Provides Graphical Displays In 2-D And *
*                    3-D With A User Menu                                   *
*****************************************************************************/

#include <graphics.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>

                        /*Function declaration */
char chkzero(),test;
char realisability(),xx;
float quadrant(float xq,float yq,float xc,float yc);
void min_interdis();

                        /* Open-Loop  Declarations */

float  Rzero[56],Izero[56],Rpole[56],Ipole[56],R_pole1[56],I_pole1[56],R_polep[56],
I_polep[56],R_zerop[56],I_zerop[56];

                        /* Closed-Loop  Declarations */

float  Rres[56],Ires[56],Rq[56],Iq[56],Rdq[56],Idq[56],R_pole[56],R_zero[56],
I_pole[56],I_zero[56],Iqq[1000],Rqq[1000],sensitive[1000];

                        /* Uncertainty & Fault Simulation  Declarations */

FRpole[56],FIpole[56],FRzero[56],FIzero[56],FRqq[200],FIqq[200],Fsens[200],Rdp[56],Idp[
56],Rdz[56],Idz[56],Rp[56],Ip[56],Rz[56],Iz[56];
int  ft=0,faultflag1=0,faultflag2=0,poleflag[56],Fstep,poleno,zerono,zeroflag[56];

                        /* Graphics  Declarations */

int  driver = DETECT, mode,zoom=50,x,y;

float  pi=3.141592654,mid,k,dk,kk=0,gain[1000],T,epsi;
int  num1,num2,RS=0,step,s,t=0;
```

```
/*************************************************************************
*              This function initialises the arrays to zero              *
*************************************************************************/
initialise()
{
int i;
for(i=0;i<66;i++)
    {
    Rzero[i]=0;     Izero[i]=0;     Rpole[i]=0;     Ipole[i]=0;
    Rres[i]=0;      Ires[i]=0;      gain[i]=0;
    Rq[i]=0;        Iq[i]=0;        Rqq[i]=0;       Iqq[i]=0;
    R_polep[i]=0;  I_polep[i]=0;  R_zerop[i]=0;   I_zerop[i]=0;
    R_pole1[i]=0;  I_pole1[i]=0;  R_pole[i]=0;    R_zero[i]=0;
    I_pole[i]=0;    I_zero[i]=0;    poleflag[i]=0;  zeroflag[i]=0;
    FRpole[i]=0;    FIpole[i]=0;    FRzero[i]=0;    FIzero[i]=0;
    FRqq[i]=0;      FIqq[i]=0;      Rdp[i]=0;       Idp[i]=0;
    Rp[i]=0;        Ip[i]=0;        Rdz[i]=0;       Idz[i]=0;
    Rz[i]=0;        Iz[i]=0;
    }
}

/*************************************************************************
* This function reads the number and the values of open-loop zeros from the keyboard *
* and stores them in real and imaginary arrays.                          *
*************************************************************************/

inputpole()
{
int i;
lable3:
printf("\nPlease enter number of poles      : ");
scanf("%d",&num1);
if(num1<1)
    {
    printf("*****************************************************\n");
    printf("Incorrect entry, type number of poles between 1 and n\n");
    goto lable3;
    }
for(i=0;i<num1;i++)
    {
    printf("please enter real part of pole p%d :      ",i);
    scanf("%f",&R_pole[i]);
    printf("please enter imaginary part of pole p%d :  ",i);
    scanf("%f",&I_pole[i]);
    R_polep[i]=R_pole[i];
    I_polep[i]=I_pole[i];
    }
}
```

```
/*************************************************************************
* This function reads  time delay, number and the values of open-loop zeros from      *
* the keyboard  and stores them in  appropriate memory locations.                      *
*************************************************************************/

inputzero()
{
int i;
lable1:
printf("\n how many steps would you like ?       ");
scanf("%d",&step);
if(step<1)
    {
    printf("*******************************************************\n");
    printf("Incorrect entry, type number of steps between 1 and n\n");
    goto lable1;
    }
lable4:
printf("\nplease enter number of zero's in the finite area of S-plane:  ");
scanf("%d",&num2);
if(num2<0)
    {
    printf("*******************************************************\n");
    printf("incorrect entry, type number of zero's between 1 and n\n ");
    goto lable4;
    }
for(i=0;i<num2;i++)
    {
    printf("please enter real part of zero z%d        :  ",i);
    scanf("%f",&R_zero[i]);
    printf("please enter imaginary part of zero z%d :  ",i);
    scanf("%f",&I_zero[i]);
    R_zerop[i]=R_zero[i];
    I_zerop[i]=I_zero[i];
    }
lablet:
printf("\nWhat is the value of time delay, Enter a value between '0' - 'n");
printf("'if there is no delay enter '0' ");
scanf("%f",&T);
if(T<0)
    {
    printf("\n*******************************************************\n");
    printf("Incorrect entry, enter time delay larger than '0' \n ");
    goto lablet;
    }
}
```

```
/*************************************************************************
* This function tests the entered system for non-physical realisable and prompts      *
*  user to re-enter the system or to quit program execution.                          *
*************************************************************************/
char realisability()
{
char ch;
int i=0,flag=0;

if(num2>num1)
    {
    printf("\n\n*******************************************************");
    printf("\n* THE SYSTEM ENTERED IS NOT PHYSICALLY REALISABLE  *");
    printf("\n* THE DENOMINATOR ORDER MUST BE HIGHER OR EQUAL TO
*");
    printf("\n* THE ORDER OF THE NUMERATOR                    *");
    printf("\n*******************************************************");

    label7:
    printf("\n\n-------------------------------------------------");
    printf("\n PRESS  ' R ' TO RETRY  OR  ' Q ' TO QUIT  :\n");
    ch=getch();
    if((ch=='r')||(ch=='R'))
            {
            return('r');
            }
    else
            {
            if((ch=='q')||(ch=='Q'))
                exit(0);
            else
                goto label7;
            }
    }

if(flag==0)
    {
    for(i=0;i<num1;i++)
            {
            if( R_pole[i]==R_pole[i+1] )
                {
                if( (I_pole[i]!=0) && (I_pole[i+1]!=0) )
                        {
                        if( fabs(I_pole[i]) !=fabs( I_pole[i+1]) )
                                {
                                if( -I_pole[i] != I_pole[i+1] )
                                    {

printf("\n\n*****************************************************************");
            printf("\n* THE SYSTEM ENTERED IS NOT PHYSICALLY REALISABLE.
*");
```

```
        printf("\n* FOR A COMPLEX CONJUGATE PAIR; REAL PARTS MUST BE
EQUAL AND    *");
        printf("\n* IMAGINARY PARTS SHOULD BE OF EQUAL VALUES WITH
OPPOSITE SIGNS *");
 printf("\n*************************************************************************");
        flag=1;
        goto label7;
                        }
                    }
                }
            }
        }
    }
}

/*************************************************************************
* This  function  reads values for uncertainty modelling and fault  simulation.          *
************************************************************************* **/
faultsim1()
{
int i;
char ch;
puts("\nDO YOU REQUIRE FAULT SIMULATION, IF YES ENTER ' Y '");
puts("                    IF NO PRESS 'ANY OTHER KEY'");
ch=getch();
if( (ch=='y')||(ch=='Y') )
    {
    flabel1:
    printf("\nAt what step should the fault occur:  ");
    scanf("%d",&Fstep);
    if( (Fstep<0) || (Fstep>step) )
        {
        puts("*****************************************************************");
        printf("\nIncorrect entry, please re-type fault occurance between");
        printf("\n ' 0 ' and %d",step);
        goto flabel1;
        }

printf("-------------------------------------------------------------");
printf("\nAre open loop poles moving ?   If yes enter 'Y'");
puts("\n                        If no  press 'any other key'");
ch=getch();
if( (ch=='y')||(ch=='Y') )
    {
    puts("-------------------------------------------------------------------------");
    for(i=0;i<num1;i++)
        {
        printf("\nPole no.(%d) = %2f + j %2f",i,R_polep[i],I_polep[i]);
        FRpole[i]=R_polep[i];
        FIpole[i]=I_polep[i];
        }
    puts("\n-------------------------------------------------------------------------");
```

```
    printf("\nHow many poles are moving ?  ");
    scanf("%d",&poleno);
    flabel2:
    for(i=0;i<num1;i++)
          {
          printf("\nHas pole no. (%d) changed ?   If yes enter ' Y '",i);
          printf("\n                          If no  press 'any other key'");
       ch=getch();
          if( (ch=='y')||(ch=='Y') )
              {
              poleflag[i]=1;
              printf("\n    please enter new real part pole no. (%d)=   ",i);
              scanf("%f",&FRpole[i]);
              printf("\nplease enter new Imaginary part pole no. (%d)=   ",i);
              scanf("%f",&FIpole[i]);
              }
          }

     for(i=0;i<num1;i++)
          {
          if( FRpole[i]==FRpole[i+1] )
              {
              if( (FIpole[i]!=0) && (FIpole[i+1]!=0) )
                    {
                    if( fabs(FIpole[i]) !=fabs( FIpole[i+1]) )
                          {
                          if( -FIpole[i] != FIpole[i+1] )
                              {

printf("\n\n*************************************************************************");
          printf("\n* THE SYSTEM ENTERED IS NOT PHYSICALLY REALISABLE.
*");
          printf("\n* FOR A COMPLEX CONJUGATE PAIR; REAL PARTS MUST BE
EQUAL AND   *");
          printf("\n* IMAGINARY PARTS SHOULD BE OF EQUAL VALUES WITH
OPPOSITE SIGNS *");

printf("\n*************************************************************************");
          goto flabel2;
                              }
                          }
                    }
              }
          }
    }
printf("-------------------------------------------------------------");
printf("\n\nAre open loop zeros moving ? If yes enter 'Y'");
puts("\n                          If no  press any other key");
ch=getch();
if( (ch=='y')||(ch=='Y') )
    {
    puts("-----------------------------------------------");
```

```
    for(i=0;i<num2;i++)
          {
          printf("     Real part zero no.(%d)= %f ",i,R_zerop[i]);
          printf("Imaginary part zero no.(%d)= %f ",i,R_zerop[i]);
          FRzero[i]=R_zerop[i];
          FIzero[i]=I_zerop[i];
          }
    puts("-------------------------------------------------");
    printf("\nHow many zeros are moving ?  ");
    scanf("%d",&zerono);
    flabel3:
    for(i=0;i<num2;i++)
          {
          printf("\nHas zero no. (%d) changed ?  If yes enter ' Y ' ");
          printf("                          If no  press any other key");
       ch=getch();
          if( (ch=='y')||(ch=='Y') )
              {
              zeroflag[i]=1;
              printf("     please enter new real part zero no. (%d)=    ",i);
              scanf("%f",&FRzero[i]);
              printf("please enter new Imaginary part zero no. (%d)=    ",i);
              scanf("%f",&FIzero[i]);
              }
          }

     for(i=0;i<num2;i++)
          {
          if( FRzero[i]==FRzero[i+1] )
              {
              if( (FIzero[i]!=0) && (FIzero[i+1]!=0) )
                    {
                    if( fabs(FIzero[i]) != fabs( FIzero[i+1]) )
                          {
                          if( -FIzero[i] != FIzero[i+1] )
                              {

printf("\n\n*************************************************************************");
          printf("\n* THE SYSTEM ENTERED IS NOT PHYSICALLY REALISABLE.
*");
          printf("\n* FOR A COMPLEX CONJUGATE PAIR; REAL PARTS MUST BE
EQUAL AND    *");
          printf("\n* IMAGINARY PARTS SHOULD BE OF EQUAL VALUES WITH
OPPOSITE SIGNS *");

printf("\n*************************************************************************");
          goto flabel2;
                              }
                          }
                    }
              }
          }
```

```
    }

   faultflag1=1;
   }
}
/*************************************************************************
* This function sets flags and assigns closed-loop and sensitivities to  *
* their corresponding designated arrays for fault simulation              *
*************************************************************************/
faultsim2()
{
int i=0,ftt;
if( (faultflag1==1) && (s==step-1) )
    {
    s=s-100;
    ft=0;
    ftt=Fstep;
    for(i=0;i<num1;i++,ft++,ftt++)
          {
          R_pole[i]=Rqq[ftt-num1];
          I_pole[i]=Iqq[ftt-num1];
          FRqq[ft]=Rqq[ftt-num1];
          FIqq[ft]=Iqq[ftt-num1];
          if(poleflag[i]==1)
              {
              Rdp[i]=(FRpole[i]-R_polep[i])/100;
              Idp[i]=(FIpole[i]-I_polep[i])/100;
              }
          Rp[i]=R_polep[i];
          Ip[i]=I_polep[i];
          if(zeroflag[i]==1)
              {
              Rdz[i]=(R_zerop[i]-FRzero[i])/100;
              Idz[i]=(I_zerop[i]-FIzero[i])/100;
              }
          Rz[i]=R_zerop[i];
          Iz[i]=I_zerop[i];
          }
    faultflag1=0;
    faultflag2=1;
    k=gain[Fstep];
    }
}
/*************************************************************************
* This function prompts user to either enter a value for dk or to select default. If *
* default is selected then the ' dkselection routine ' is executed which invokes      *
* ' min_interdis routine ' automatically calculates a value for dk according to the   *
* open-loop pole/zero pattern.                                                        *
*************************************************************************/
delta_k()
{
char ch;
```

```
lable2:
printf("\nPress 'A' to select dk automatically  (default)    'OR'");
printf("\n\npress 'E' to enter a value for Magnification/attenuation ");
printf("of the default value of dk  :  ");
ch=getch();
if((ch=='e')||(ch=='E'))
    {
    lable2a:
    printf("\nType a value for magnif/atten factor between 0.1 and 80:  ");
    scanf("%f",&epsi);
    if((epsi<=0.1)||(epsi>80))
          {
          printf("\n\n\n\n*************************************************************");
          printf("\nINCORRECT ENTRY, TRY AGAIN BY TYPING DK BETWEEN 0
AND 1\n");
          printf("*************************************************************\n");
          ch=getch();
          goto lable2a;
          }
    }
else
    {
    if((ch=='a')||(ch=='A'))
          {
          epsi=1;
          }
    else
          {

printf("\n\n\n\n\n\n*************************************************************");
          printf("\n      INCORRECT ENTRY     PLEASE TRY AGAIN        ");
          printf("\n*************************************************************\n");
          ch=getch();
          goto lable2;
          }
    }
}
/*****************************************************************************
* This function is invoked when  multiple open-loop  poles /zeros are detected .  It      *
* calculates new co-ordinates for each C.L. pole merging from the multiple point        *
*****************************************************************************/
multiplepole()
{
int count1,count2,flag2[256],counter=0,a,b=0,c=0,d,g=0,h=0,i=0,j=0,j1,k=0,
flag1=0,l,m,tn,frac2=0,flag3=0,odd;
float Rgroup[256],Igroup[256],groupno[256],skip[256],q,xq,yq,xc,yc,
Rnewq[256],Inewq[256],zerofi=0,polefi=0,fi,fi1,fi2,frac1,theta,epsi1,
mindistance;
mindistance=mid;
epsi1=mindistance*0.0001;
for(a=0;a<10;a++)
    {
```

```
    Rgroup[a]=0; Igroup[a]=0; flag2[a]=0; skip[a]=0;
    groupno[a]=0,Rnewq[a]=0,Inewq[a]=0;
    }
while(counter<num1-1)
    {
    a=i+1,count1=0;
    while(count1<num1-(b+1))
          {
          if( ((R_pole[i]-R_pole[a])==0)&&((I_pole[i]-I_pole[a])==0))
              {
              d=0,count2=0,flag3=1;
              while(count2<c)
                    {
                    if(i==skip[d])
                          {
                          flag1=1;
                          }
                    count2++,d++;
                    }
              if(flag1==1)
                    {
                    a++,flag1=0;
                    }
              else
                    {
                    if(g==0)
                          {
                          Rgroup[k]=R_pole[i];
                          Igroup[k]=I_pole[i];
                          flag2[h]=i;
                          k++,h++;
                          }
                    Rgroup[k]=R_pole[a];
                    Igroup[k]=I_pole[a];
                    skip[c]=a;
                    flag2[h]=a;
                    g++,k++,c++,h++;
                    }
              } /*if ( ) */
          count1++,a++;
          } /* while count1 */
    if(g!=0)
          {
          groupno[j]=g+1;
          g=0,j++;
          }
    i++,counter++,b++;
    } /* while counter */
a=0,b=0,odd=1,counter=0,j1=0;
while(counter<k)
    {
    xq=Rgroup[a]+epsi1;
```

```
    yq=Igroup[a];
    count1=0,i=0;
    while(count1<num2)
        {
        xc=R_zero[i];
        yc=I_zero[i];
        fi1=quadrant(xq,yq,xc,yc);
        zerofi=zerofi+fi1;
        i++,count1++;
        }
    count1=0,i=0;
    while(count1<num1)
        {
        xc=R_pole[i];
        yc=I_pole[i];
        fi2=quadrant(xq,yq,xc,yc);
        polefi=polefi+fi2;
        i++,count1++;
        }
    frac1=(zerofi-polefi)/2*pi;
    frac2=frac1;
    fi=frac1-frac2;
    theta=(odd*pi+fi)/groupno[j1];
    Rnewq[b]=Rgroup[b]+epsi1*cos(theta);
    Inewq[b]=Igroup[b]+epsi1*sin(theta);
    counter++,a++;
    odd=odd+2,b++;
    if(j1<j-1)
        j1++;
    }
if(flag3!=0)
    {
    counter=0,l=0,a=0;
    while(counter<h)
        {
        count1=0;
        while(count1<num1)
            {
            if(flag2[l]==count1)
                {
                m=flag2[l];
                R_pole[m]=Rnewq[a];
                I_pole[m]=Inewq[a];
                a++,count1=num1;
                }
            count1++;
            }
        counter++,l++;
        }
    }
}
```

```
/*************************************************************************
* This function displays the steps of the calculation cycle and a messages on the      *
*  screen while  calculation in progress                                                *
*************************************************************************/
printscreen()
{
printf("\n==========================================================
====");
printf("\n       please wait calculation in progress              ");
printf("\n==========================================================
====\n");
printf("STEP : ");
}
/*************************************************************************
* This function substitutes for s at each pole, multiplies numerator                   *
* brackets i.e. zero's for each pole separately and stores the results                  *
* in Rzero[] (real parts) and Izero[] (imaginary parts).                                *
*************************************************************************/

reszero()

{
int i,count,a=0,counter=0,j=0,d=0,m=0;
float REAL1[56],IMJ1[56],R1[56],I1[56],Rdelay,Idelay,qR,qI;
if(num2>0)
    {
    while(counter<num1)
          {
          i=0,count=0;
          while(count<num2)
              {
              R1[j]=R_pole[a]-R_zero[i];
              I1[j]=I_pole[a]-I_zero[i];
              i++,j++,count++;
              }
          a++,counter++;
          }
    counter=0;
    while(counter<num1)
          {
          count=1;
          while(count<num2)
              {
                  REAL1[d+1]=(R1[d]*R1[d+1])-(I1[d]*I1[d+1]);
                  IMJ1[d+1]=(R1[d]*I1[d+1])+(I1[d]*R1[d+1]);
                  R1[d+1]=REAL1[d+1];
                  I1[d+1]=IMJ1[d+1];
                  d++,count++;
                  }
          Rzero[m]=R1[d];
          Izero[m]=I1[d];
          m++,d++,counter++;
```

```
        }
    }
    else
    {
    for(i=0;i<num1+1;i++)
        {
        Rzero[i]=1;
        Izero[i]=0;
        }
    }       /* Multiplies time delay to numerator */
if(T>0)
    {
    i=0;
    while(i<num1)
        {
        qR=R_pole[i]*T;
        qI=I_pole[i]*T;
        Rdelay=exp(qR)*cos(qI);
        Idelay=-(exp(qR)*sin(qI));
        Rzero[i]=Rdelay*Rzero[i]-Idelay*Izero[i];
        Izero[i]=Rdelay*Izero[i]+Idelay*Rzero[i];
        i++;
        }
    }
}
/*************************************************************************
* This function substitutes for s at each pole, multiplies denominator  *
* brackets i.e. pole's for each pole separately and stores the results   *
* in Rpole[](real parts) and Ipole[] (imaginary parts).                  *
*************************************************************************/
respole()
{
int i,count,a=0,counter=0,j=0,c=0,n=0;
float REAL[56],IMJ[56],R[56],I[56];
if(num1>=2)
    {
    while(counter<num1)
        {
        i=0,count=0;
        while(count<num1-1)
            {
            if(a==i)
                i++;
            else
                {
                R[j]=R_pole[a]-R_pole[i];
                I[j]=I_pole[a]-I_pole[i];
                i++,j++,count++;
                }
            }
        a++,counter++;
        }
```

```
    if(num1==2)
            {
            Rpole[0]=R[0];
            Rpole[1]=R[1];
            Ipole[0]=I[0];
            Ipole[1]=I[1];
            }
    else
            {
            counter=0;
            while(counter<num1)
                {
                count=1;
                while(count<num1-1)
                        {
                        REAL[c+1]=(R[c]*R[c+1])-(I[c]*I[c+1]);
                        IMJ[c+1]=(R[c]*I[c+1])+(I[c]*R[c+1]);
                        R[c+1]=REAL[c+1];
                        I[c+1]=IMJ[c+1];
                        c++,count++;
                        }
                Rpole[n]=R[c];
                Ipole[n]=I[c];
                n++,counter++,c++;
                }
            }
    }
else            /*In the case of a single pole Residue is = 1+j0  */
    {
    for(i=0;i<num1+1;i++)
            {
            Rpole[i]=1;
            Ipole[i]=0;
            }
    }
}
/*************************************************************************
* This function calculates the residues by dividing the previously calculated zero and  *
*  pole values for all branches.                                                        *
*************************************************************************/

residue()
{
int c=0,count=0;

while(count<num1)
    {
    Rres[RS]=((Rzero[c]*Rpole[c])+(Izero[c]*Ipole[c]))/
                        ((Rpole[c]*Rpole[c])+(Ipole[c]*Ipole[c]));
    Ires[RS]=((Izero[c]*Rpole[c])-(Ipole[c]*Rzero[c]))/
                        ((Rpole[c]*Rpole[c])+(Ipole[c]*Ipole[c]));
    c++,RS++,count++;
```

```
    }
}
/*************************************************************************
* This function automatically calculates the minimum inter-distances between poles    *
*  and zeros. Tests for floating point error and takes steps to prevent it occurrence.    *
*************************************************************************/
void min_interdis()
{
int i=0,a,k=0,j=0,m=0,counter=0,count,pt=0,n,g;
float R_diff_pole[256],I_diff_pole[256],R_diff_zero[256],I_diff_zero[256]
,R_diff_zp[256],I_diff_zp[256],sq_diff[1024],temp,sqmid;
while(counter<num1-1) /******** Pole-Differnces Caculation ************/
    {
    a=i+1;count=0;
    while(count<num1-(i+1))
            {
            R_diff_pole[k]=R_polep[i]-R_polep[a];
            I_diff_pole[k]=I_polep[i]-I_polep[a];
            a++,count++,k++;
            }
    i++,counter++;
    }
for(i=0;i<10;i++)
    sq_diff[i]=0;

count=0,i=0;
while(count<k)
    {
    sq_diff[pt]=(R_diff_pole[i]*R_diff_pole[i])+(I_diff_pole[i]*I_diff_pole[i]);
    i++,count++,pt++;
    }
if(num2>0)
    {
    i=0,counter=0;     /********* Zero-Differences Calculation **********/
    while(counter<num2-1)
            {
            a=i+1;count=0;
            while(count<num2-(i+1))
                {
                R_diff_zero[j]=R_zerop[i]-R_zerop[a];
                I_diff_zero[j]=I_zerop[i]-I_zerop[a];
                a++,count++,j++;
                }
            i++,counter++;
            }
            count=0,i=0;
            while(count<j)
                {
                sq_diff[pt]=(R_diff_zero[i]*R_diff_zero[i])+
                                (I_diff_zero[i]*I_diff_zero[i]);
                i++,count++,pt++;
                }
```

```
                    /*********Zero-Pole Differences Calculation***********/
    i=0,count=0,counter=0;
    while(counter<num1)
          {
          a=0,count=0;
          while(count<num2)
             {
             R_diff_zp[m]=R_polep[i]-R_zerop[a];
             I_diff_zp[m]=I_polep[i]-I_zerop[a];
             a++,m++,count++;
             }
          i++,counter++;
          }
    count=0,i=0;
    while(count<m)
          {
          sq_diff[pt]=(R_diff_zp[i]*R_diff_zp[i])+(I_diff_zp[i]*I_diff_zp[i]);
          i++,count++,pt++;
          }
    }
                                /**** Sort Sq-Diff[] In Order From *****/
n=pt-1;                         /**** Smallest To Largest Value    *****/
for(g=n/2; g>0; g/=2)
    {
    for(i=g; i<n;i++)
          {
          for(j=i-g; (j>=0 && sq_diff[j]>sq_diff[j+g]);j-=g)
             {
             temp=sq_diff[j];
             sq_diff[j]=sq_diff[j+g];
             sq_diff[j+g]=temp;
             }
          }
    }
i=0,count=0;          /***********Square Min Inter Distance **************/
sqmid= sq_diff[i];
if(sqmid==0 )         /* If Min. Interdistance Sqmid=0, And No Previous*/
                      /*  Value Exists Then Sqmid=0.1. But If More Than Two*/
                      /* Interdistances Exist Then The Next Smallest Value */
                      /* Is Selected. This Is Repeated For All Sqmids.If */
                      /* Still Sqmid=0 Ie. For Multiple Pole Then Sqmid=0.1 */
    {
    if(pt<2)
          {
          sqmid=0.1;
          }
    else
          {
          while(count<pt-1)
             {
             if(sqmid==0)
                   {
```

```
                    sqmid=sq_diff[i+1];
                    }
                i++,count++;
                }
            if(sqmid==0)
                sqmid=0.1 ;
            }
    }
            mid=sqrt(sqmid);
}

/*************************************************************************
* This function calculates the dk used in the program cycle by dividing the min inter  *
*  distances of poles and zeros by the largest residue of  the previous step. This      *
*  provides equal step calculations of closed-loop pole positions along each branch.   *
*************************************************************************/
dk_cycle()
{
int i=0;
float res[56];
if(faultflag2==1)
    {
    ;
    }
else
    {
    while(i<num1)
            {
            res[i]=sqrt(Rres[i]*Rres[i]+Ires[i]*Ires[i]);
            i++;
            }
    i=0;
    while(i<num1-1)
            {
            if(res[i]>res[i+1])
                res[i+1]=res[i];
            i++;        /* square of min_interdistance/largest res. */
            }
    dk=(epsi*0.01*mid)/res[i];
    }
}

/*************************************************************************
* This function calculates the closed-loop pole of each branch of step one  calculation  *
*************************************************************************/
pole_q()
{
int i=0,count=0;
while(count<num1)
    {
    Rq[i]=R_pole[i]+dk*(-Rres[i]);
    Iq[i]=I_pole[i]+dk*(-Ires[i]);
```

```
    i++,count++;
    }
}
/*************************************************************************
* This function stores the calculated closed-loop poles of step one calculation in the *
* arrays for ready for step two calculation.                                *
*************************************************************************/
newpole1()
{
int i=0,count=0;
while(count<num1)
    {
    R_pole1[i]=R_pole[i];
    I_pole1[i]=I_pole[i];
    R_pole[i]=Rq[i];
    I_pole[i]=Iq[i];
    i++,count++;
    }
}
/*************************************************************************
* This function, in the no-fault mode, performs a correction step for greater accuracy. *
* It takes the average value of residues calculated over step one and two, adds the *
* result first time round to original open loop pole and from then          *
* onward to the calculated closed loop pole i.e. q03=(dk0+dk1)/2=(-R01)+(-R02)/2 *
* In the fault mode, new closed-loop system values are calculated           *
*************************************************************************/
cpole()
{
int i=0,counter=0,a=0,b=0;
float Rresidue[56],Iresidue[56],RDq[56],IDq[56],
Rdq1[56],Idq1[56],Rdq2[56],Idq2[56],A,B,C,D;
for(i=0;i<56;i++)
    {
    Rresidue[i]=0;
    Iresidue[i]=0;
    Rdq1[i]=0;
    Idq1[i]=0;
    Rdq2[i]=0;
    Idq2[i]=0;
    RDq[i]=0;
    IDq[i]=0;
    }

for(i=0;i<num1;i++)
    {
    Rresidue[i]=((-Rres[i])+(-Rres[i+num1]))/2;
    Iresidue[i]=((-Ires[i])+(-Ires[i+num1]))/2;
    }
if(faultflag2==1)
    {
    while(counter<num1)
        {
```

```
        for(i=0;i<num1;i++)
            {
            if (poleflag[i]==1)
                    {
                    A=FRqq[ft-num1]-Rp[i];
                    B=FIqq[ft-num1]+Ip[i];
                    C=Rresidue[a]*Rdp[i]-Iresidue[a]*Idp[i];
                    D=Rresidue[a]*Idp[i]+Iresidue[a]*Rdp[i];
                    RDq[i]=k*( C*A + D*B )/( A*A + B*B );
                    IDq[i]=k*( D*A - C*B )/( A*A + B*B );
                    }
            }
        for(i=0;i<num1;i++)
            {
            Rdq1[b]=Rdq1[b] + RDq[i];
            Idq1[b]=Idq1[b] + IDq[i];
            }

        for(i=0;i<num2;i++)
            {
            if(zeroflag[i]==1)
                    {
                    A=Rz[i]-FRqq[ft-num1];
                    B=Iz[i]-FIqq[ft-num1];
                    C=Rresidue[a]*Rdz[i]-Iresidue[a]*Idz[i];
                    D=Rresidue[a]*Idz[i]+Iresidue[a]*Rdz[i];
                    RDq[i]=k*( C*A + D*B )/( A*A + B*B );
                    IDq[i]=k*( D*A - C*B )/( A*A + B*B );
                    }
            }
        for(i=0;i<num2;i++)
            {
            Rdq2[b]=Rdq2[b] + RDq[i];
            Idq2[b]=Idq2[b] + IDq[i];
            }
        FRqq[ft]=FRqq[ft-num1]+Rdq1[b]+Rdq2[b];
        FIqq[ft]=FIqq[ft-num1]+Idq1[b]+Idq2[b];
        Fsens[ft]=gain[ft]*sqrt(Rresidue[i]*Rresidue[i]+Iresidue[i]*Iresidue[i]);
        for(i=0;i<num1;i++)
            {
            Rp[i]=Rp[i]+Rdp[i];
            Ip[i]=Ip[i]+Idp[i];
            }
        ft++,a++,b++,counter++;
        }
    }

else
    {
    i=0;
    kk=kk+dk;
    while(i<num1)
```

```
        {
        Rdq[i]=dk*Rresidue[i];
        Idq[i]=dk*Iresidue[i];
        Rqq[t]=R_pole1[i]+Rdq[i];
        Iqq[t]=I_pole1[i]+Idq[i];
        gain[t]=kk;
        sensitive[t]=gain[t]*sqrt(Rresidue[i]*Rresidue[i]+Iresidue[i]*Iresidue[i]);
        i++,t++;
        }
    }
}
/*************************************************************************
* This function detects any number of closed-loop poles approaching  breakaway point *
* and calculates  new q co-ordinates emerging it. The calculations hence continues on *
*  new loci from the breakaway point.                                    *
*************************************************************************/
branch()
{
int count1=0,count2,flag2[256],counter=0,a,b=0,c=0,d,g=0,h=0,i=0,j=0,k=0,
flag1=0,l,m,tn,flag3=0;
float Rgroup[256],Igroup[256],groupno[256],skip[256],RB,IB,RBR[256],
IBR[256],Radius[256],xc,yc,Entryphi[256],Exitphi[256],Rq_b[256],Iq_b[256],
Rdq_b[256],Idq_b[256],Rnewq[256],Inewq[256],sq_diff_q,sq_dq1,sq_dq2,sn,cs;
tn=t-num1;
while(count1<num1)
    {
    Rq_b[i]=Rqq[tn];
    Iq_b[i]=Iqq[tn];
    Rdq_b[i]=Rdq[i];
    Idq_b[i]=Idq[i];
    i++,tn++,count1++;
    }
i=0;
while(counter<num1-1)
    {
    a=i+1,count1=0;
    while(count1<num1-(b+1))
        {
        sq_diff_q=(pow((Rq_b[i]-Rq_b[a]),2)+pow((Iq_b[i]-Iq_b[a]),2));
        sq_dq1=(Rdq_b[i]*Rdq_b[i]+Idq_b[i]*Idq_b[i]);
        sq_dq2=(Rdq_b[a]*Rdq_b[a]+Idq_b[a]*Idq_b[a]);
        if( (sq_diff_q<=sq_dq1)||(sq_diff_q<=sq_dq2) )
            {
            d=0,count2=0,flag3=1;
            while(count2<c)
                {
                if(i==skip[d])
                    {
                    flag1=1;
                    }
                count2++,d++;
                }
```

```
                if(flag1==1)
                        {
                        a++,flag1=0;
                        }
                else
                        {
                        if(g==0)
                                {
                                Rgroup[k]=Rq_b[i];
                                Igroup[k]=Iq_b[i];
                                flag2[h]=i;
                                k++,h++;
                                }
                        Rgroup[k]=Rq_b[a];
                        Igroup[k]=Iq_b[a];
                        skip[c]=a;
                        flag2[h]=a;
                        g++,k++,c++,h++;
                        }
                } /*if pow( ) */
            count1++,a++;
            } /* while count1 */
    if(g!=0)
            {
            groupno[j]=g+1;
            g=0,j++;
            }
    i++,counter++,b++;
    } /* while counter */
counter=0,i=0,a=0,j=0;
while(counter<k)
    {
    count1=0,RB=0,IB=0;
    while(count1<groupno[j])
            {
            RB=RB+Rgroup[i];
            IB=IB+Igroup[i];
            i++,count1++,counter++;
            }
    RBR[a]=RB/groupno[j];
    IBR[a]=IB/groupno[j];
    Radius[a]=sqrt(pow((RBR[a]-Rgroup[counter-1]),2)+
                                    pow((IBR[a]-Igroup[counter-1]),2));
    a++,j++;
    }
counter=0,a=0,i=0,j=0;
while(counter<k)
    {
    count1=0;
    while(count1<groupno[i])
            {
            xc=RBR[i];
```

```
            yc=IBR[i];
            Entryphi[a]=quadrant(Rgroup[j],Igroup[j],xc,yc);
            Exitphi[a]=Entryphi[a]+(pi/groupno[i]);
            cs=cos(Exitphi[a]);
            Rnewq[a]=RBR[i]+(Radius[i]*cs);
            sn=sin(Exitphi[a]);
            Inewq[a]=IBR[i]+(Radius[i]*sn);
            a++,j++,counter++,count1++;
            }
    i++;
    }
if(flag3!=0)
    {
    counter=0,l=0,a=0;
    while(counter<h)
            {
            count1=0;
            while(count1<num1)
                {
                if(flag2[l]==count1)
                        {
                        m=flag2[l];
                        Rq_b[m]=Rnewq[a];
                        Iq_b[m]=Inewq[a];
                        a++,count1=num1;
                        }
                count1++;
                }
            counter++,l++;
            }
    count1=0,tn=t-num1;
    while(count1<num1)
            {
        Rqq[tn]=Rq_b[count1];
            Iqq[tn]=Iq_b[count1];
            count1++,tn++;
            }
    }
}
/*****************************************************************************
* This function  tests the accuracy of the calculated closed-loop system by calculating *
*  the error angle due from closed loop poles and zeros. If the error is greater than a  *
* tolerance of 10^-6 then the program repeats the calculation with half the value of dk,  *
*  until the required accuracy is achieved.                                              *
*****************************************************************************/
chkangle()
{
int count=0,counting=0,i=0,frac2;
float xq,yq,xc,yc,zerofi=0,polefi=0,fi1,fi2,fraction,error,
frac1,netangle,epsi1=0.001*dk;
if(counting==10)
    {
```

```
    if(faultflag2==1)
            {
            xq=FRqq[ft-num1];
            yq=FIqq[ft-num1];
            }
    else
            {
            xq=Rqq[t-num1];
            yq=Iqq[t-num1];
            }
    while(count<num2)
            {
            xc=R_zerop[i];
            yc=I_zerop[i];
            fi1=quadrant(xq,yq,xc,yc);
            zerofi=zerofi+fi1;
            i++,count++;
            }
    count=0,i=0;
    while(count<num1)
            {
            xc=R_polep[i];
            yc=I_polep[i];
            fi2=quadrant(xq,yq,xc,yc);
            polefi=polefi+fi2;
            i++,count++;
            }
    netangle=zerofi-polefi;
    frac1=fabs(netangle/(2*pi));
    frac2=frac1;
    fraction=frac1-frac2;
    error=fabs(1-2*fraction);
    if(error>=epsi1)
            {
            RS=0;
            if(faultflag2==1)
                {
                k=k/2;
                ft=ft-(10*num1)+num1;
                }
            else
                {
                dk=dk/2;
                t=t-(10*num1)+num1;
                }
            s=s-10;
            }
    counting=0;
    }
counting++;
}
```

```
/*************************************************************************
* This function calculates all angles of vectors drawn from each open    *
* loop pole and zero towards a current closed loop q. Then it will       *
* determines which quadrant the angle lies in.                           *
*************************************************************************/
float quadrant(float xq,float yq,float xc,float yc)
{
float dx,epsi1=0.0001,epsi2,angle;
epsi2=epsi1*dk;
dx=fabs(xq-xc);
if(dx<=epsi2)
   {
   if(yq>=yc)                  /**** 90 Degrees ****/
         {
         return(pi/2);
         }
   else if(yq<yc)               /**** 270 Degrees ****/
         {
         return(pi*1.5);
         }
   }
else if((yq>=yc) && (xq>xc))          /**** 1st quadrant ****/
         {
         angle=atan((yq-yc)/(dx));
         return(angle);
         }
   else if((yq>=yc) && (xq<xc))      /**** 2nd quadrant ****/
            {

            angle=atan((yq-yc)/(dx));
            return(pi-angle);
            }
         else if((yq<yc) && (xq<xc))  /***** 3rd quadrant ****/
                  {
                  angle=atan((yc-yq)/(dx));
                  return(pi+angle);
                  }
            else if((yq<yc) && (xq>xc))     /**** 4th quadrant ****/
                     {
                     angle=atan((yc-yq)/(dx));
                     return(2*pi-angle);
                     }
}

/*************************************************************************
* this function stores the calculated closed loop poles of step three back in arrays    *
*  ready for step one of the next calculation cycle.                                     *
*************************************************************************/
newpole2()
{
int i=0,tt,ft2;
tt=t;
```

```
ft2=ft;
while(i<num1)
    {
    if(faultflag2==1)
            {
            R_pole[i]=FRqq[ft2-num1];
            I_pole[i]=FIqq[ft2-num1];
            ft2++;
            }
    else
            {
            R_pole[i]=Rqq[(tt-(num1))];
            I_pole[i]=Iqq[(tt-(num1))];
            tt++;
            }
    i++;
    }
RS=0;
}
/*************************************************************************
* This function saves  3-D data on a data file   " a:DD.D "                *
*************************************************************************/
data_out()
{
FILE  *DD;
int i=0;
float x,y,z;

DD=fopen("c:DD.D","wt");
if( (DD==NULL) )
    {
    fprintf(stderr,"Cannot open input file.\n");
    return(1);
    }
while(i<t)
    {
          /******************* 3-D sensitivity data ***************/
    x=Rqq[i];
    y=Iqq[i];
    z=sensitive[i];
    fprintf(DD,"\n%d   %f    %f    %f",i,x,y,z);
    i++;
    }
}

/*************************************************************************
* This function displays the calculated closed-loop poles for all branches by plotting a  *
*  pixel at each position of Rqq[],Iqq[]. Starting form location ' 0 ' in S domain.        *
*************************************************************************/
plot()
{
float x,y;
```

```
int count=0,color,g;
g=graphresult();
if(g==0)
    {
    setcolor(15);
    setlinestyle(0,0,3);
    while(count<t)
            {
            x=400+Rqq[count]*zoom;
            y=200-Iqq[count]*zoom;
           /*  if((count-num1)<=0)
                moveto(x,y);
            else
                moveto((400+Rqq[count-num1]*zoom),(200-Iqq[count-num1]*zoom));
            lineto(x,y);  */
            putpixel(x,y,15);

            count++;
            }
     }
else
    printf("graphic error occured,unable to plot");

}

/*************************************************************************
* This function  plots the calculated closed-loop poles under the fault condition.      *
*************************************************************************/
faultsimplot()
{
int i,g;
float x,y;
if(faultflag2==1)
    {
    g=graphresult();
    if(g==0)
            {
            disbackgn();
            setcolor(14);
            outtextxy(60,30,"FAULT PLOT ");
            setlinestyle(0,0,1);
            i=0;
            while(i<ft)
                {
                x=400+FRqq[i]*zoom;
                y=200+FIqq[i]*zoom;
                putpixel(x,y,14);
                i++;
                }
            }
    graphics();
    }
```

```
else
    {
    cleardevice();
    outtextxy(120,150,"NO FAULT HAS BEEN DETECTED ");
    graphics();
    }
}
/*****************************************************************************
* This function displays the background for S-pane drawing real and imaginary axis,  *
*  labels, marks etc.                                                                 *
*****************************************************************************/
disbackgn()
{
int i=0,count=0,xoffset=400,yoffset=200;
float px[56],py[56],zx[56],zy[56],x1,y1,gap;
x=getmaxx();
y=getmaxy();
setbkcolor(0);
setcolor(10);
rectangle(50,20,x-5,y-120);
x1=50;
y1=y-120;
gap=(x-50)/20;
while(x1<x)
    {
    line(x1,y1-4,x1,y1+4);
    x1=x1+gap;
    }
x1=50;
y1=20;
gap=(y-120)/20;
while(y1<y-120)
    {
    line(x1-4,y1,x1+4,y1);
    y1=y1+gap;
    }
outtextxy(x-100,40,"S DOMAIN");
outtextxy(x-39,y-110,"REAL");
settextstyle(0,1,1);
outtextxy(10,10,"IMAGINARY");
settextstyle(3,0,1);
setcolor(10);
line(400,20,400,y-120);
/*line(50,200,x,200); */
setcolor(11);
/* outtextxy(x-480,0,"SENSITIVITY BASED  TECHNIQUE");*/
setcolor(12);
while(count<num1)
    {
    px[i]=xoffset+R_polep[i]*zoom;
    py[i]=yoffset-I_polep[i]*zoom;
    line(px[i]-5,py[i]-5,px[i]+5,py[i]+5);
```

```
    line(px[i]-5,py[i]+5,px[i]+5,py[i]-5);
    i++,count++;
    }
i=0,count=0;
while(count<num2)
    {
    zx[i]=xoffset+R_zerop[i]*zoom;
    zy[i]=yoffset-I_zerop[i]*zoom;
    circle(zx[i],zy[i],5);
    i++,count++;
    }

}
/*****************************************************************************
* This function displays the commands menu ready for  user selection                *
*****************************************************************************/
graphics()
{
char ch;
short *done=0;
setlinestyle(0,0,1);
setcolor(11);
outtextxy(60,380,"PLEASE SELECT FROM MENU ==>");
setcolor(10);
rectangle(50,400,x-5,y);
settextstyle(3,0,1);
outtextxy(60,400,"E: Exit program");
outtextxy(350,400,"Z: zoom-in and zoom-out ");
outtextxy(350,420,"F: Plot sensitivity under fault");
outtextxy(60,420,"D: Save graphic data in a file");
outtextxy(350,440,"S: Plot Sensitivity");
outtextxy(60,440,"P: plot root locus");
outtextxy(60,460,"R: Plot root locus under fault");
moveto(400,380);
do
    {
    ch=getch();
    switch(ch)
            {
            case 'e':
            case 'E': _exit(0);break;
            case 'd':
            case 'D': data_out();break;
            case 'p':
            case 'P': replot();break;
            case 's':
            case 'S': sensitivity();break;
            case 'z':
            case 'Z': zooming();break;
            case 'f':
            case 'F': faultsensitivity();
            case 'r':
```

```
            case 'R': faultsimplot();
            }
    }while(!*done);
    labelr:;
}
/*************************************************************************
* This function plots the 3-d sensitivity profiles of all branches on the  screen.          *
*************************************************************************/
sensitivity()
{
float x,y,x1,y1,theta=-0.75,phi=0.5,sen;
int count=0,xoffset=500,yoffset=300;
cleardevice();
x=getmaxx();
/* y=getmaxy(); */
/*-------------------------------------------------------------
printf("\n\n\nFOR 3-DIMENTIONAL PLOTTING ANGLES 'theta' & 'phi' ARE
REQUIRED");
printf("\n\nPLEASE ENTER ANGLE 'theta', TYPICALLY BETWEEN 20..40 ==>");
scanf("%f",&theta);
printf("\nPLEASE ENTER ANGLE 'phi', TYPICALLY BETWEEN 40..80 DEGREES
==>");
scanf("%f",&phi);
setcolor(15);
cleardevice();
rectangle(50,20,x,y-120);   --------------------------------*/

settextstyle(3,0,1);
outtextxy(x-480,0,"THREE-DIMENTIONAL SENSITIVITY PLOT");
if(graphresult()==0)
    {
    count=0;
    while(count<t)
            {
            x1=xoffset+( Iqq[count]*sin(phi)+Rqq[count]*cos(theta) )*zoom;
            y1=yoffset-( Iqq[count]*cos(phi)+Rqq[count]*sin(theta) )*zoom;
            putpixel(x1,y1,15);
            sen=sensitive[count]*zoom;
            putpixel(x1,y1-sen,12);
            setcolor(14);
            setlinestyle(1,0,1);
            line(x1,y1-4,x1,y1-sen-4);
            count++;
            }
            graphics();
    }
else
    puts("Graphic error occured");
}
```

```
/*****************************************************************************
* This function  plots 3-d sensitivity profiles under fault conditions                *
*****************************************************************************/
faultsensitivity()
{
float x,x1,y1,theta=-0.75,phi=0.5,sen;
int count=0,xoffset=400,yoffset=200;
cleardevice();
if(faultflag2==1)
   {
  /*  sensitivity(); */
   x=getmaxx();
   settextstyle(3,0,1);
   outtextxy(x-520,0,"THREE-DIMENTIONAL   SENSITIVITY   PLOT   UNDER
FAULT");
   if(graphresult()==0)
         {
         setcolor(15);
         count=0;
         while(count<ft)
            {
            x1=xoffset+( FIqq[count]*sin(phi)+FRqq[count]*cos(theta) )*zoom;
            y1=yoffset-( FIqq[count]*cos(phi)+FRqq[count]*sin(theta) )*zoom;
            sen=Fsens[count]*50;
            putpixel(x1,y1-sen,4);
            setcolor(4);
            setlinestyle(1,0,1);
            line(x1,y1-4,x1,y1-sen-14);
            count++;
            }
         }
   else
         puts("Graphic error occured");
   graphics();
   }
else
   {
   cleardevice();
   outtextxy(120,150,"NO FAULT HAS BEEN DETECTED ");
   graphics();
   }
}
/*****************************************************************************
* This function clears existing graphic screen and re-displays background and closed-  *
*  loop positions.                                                                    *
*****************************************************************************/
replot()
{
cleardevice();
disbackgn();
plot();
graphics();
```

```
}
/*************************************************************************
* This function allows he user to  zoom into the required display window                *
*************************************************************************/
zooming()
{
char ch;
outtextxy(140,200,"would you like to change the zoom factor");
outtextxy(140,220,"If yes press 'Y' otherwise press any other keythe");
ch=getch();
if( (ch=='y') ||(ch=='Y') )
    {
    outtextxy(140,250,"Please enter zooming factor:   ");
    scanf("%f",&zoom);
    cleardevice();
    disbackgn();
    plot();
    graphics();
    }
ch='n';
}

/*****************************Main Program*********************************/
main()
{
initialise();
label6:
inputzero();
inputpole();
xx=realisability();
if(xx=='r')
    goto label6;
min_interdis();
faultsim1();
delta_k();
multiplepole();
printscreen();
s=0;
while(s<step)
    {
    respole();
    reszero();
    residue();
    dk_cycle();
    pole_q();
    newpole1();
    respole();
    reszero();
    residue();
    cpole();
    branch();
    chkangle();
```

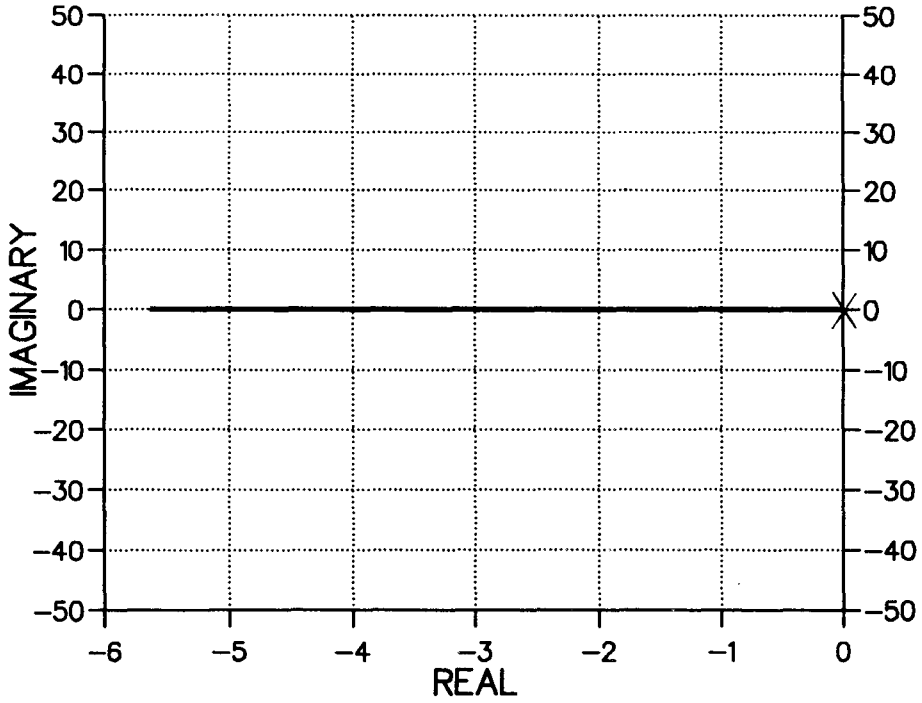
```
    newpole2();
    cprintf("  %d",s);
    faultsim2();
    s++;
    }/*step*/
label8:
initgraph(&driver, &mode,"");
replot();
}/*main*/
```

□

Appendix B

Knowledge-Base Graphic Sensitivity Classification (KBSC) Of Feed-Back Systems With Deterministic Plants

ROOT LOCUS



3-D SENSITIVITY PROFILE

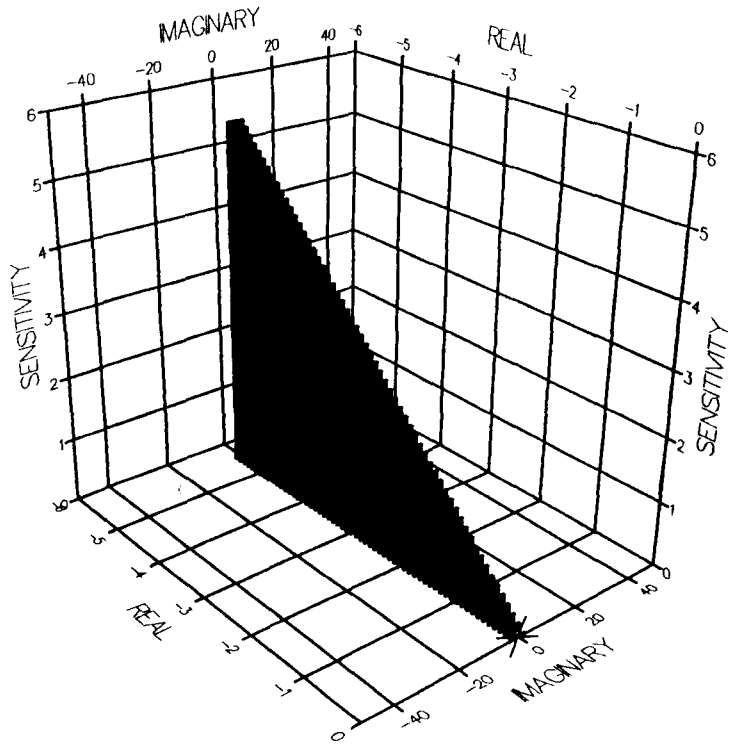
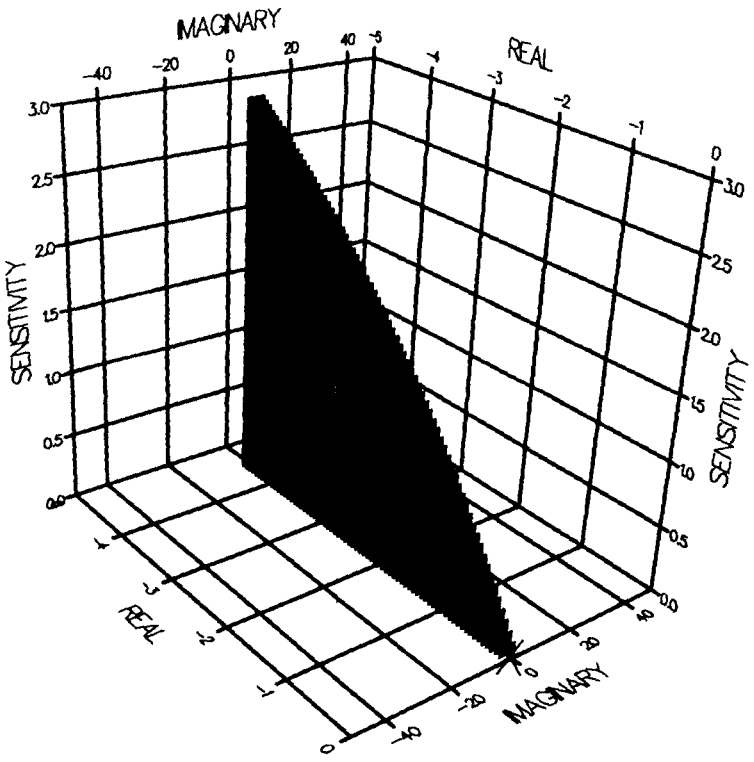
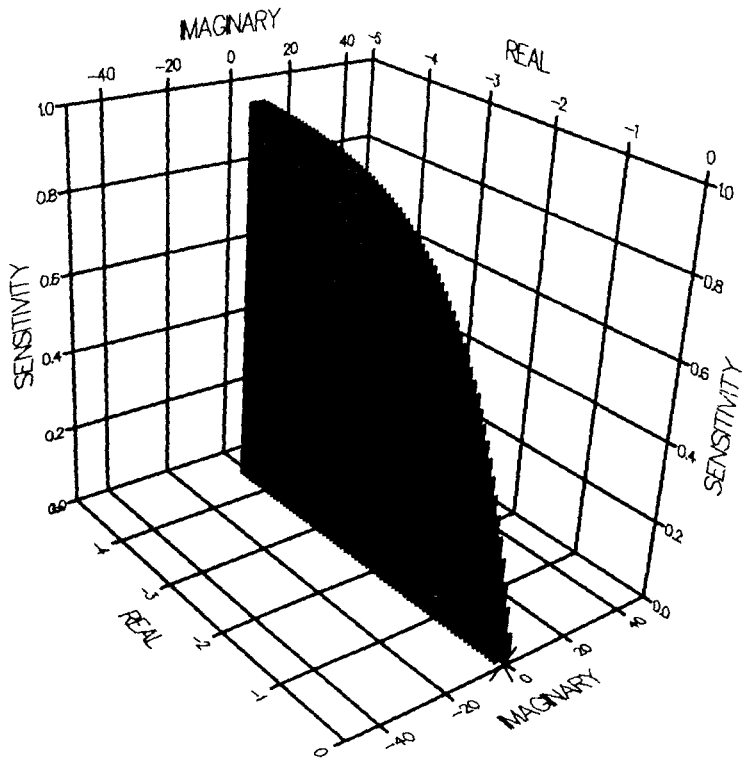


FIGURE 4.1A FIRST-ORDER PLANT

3-D SENSITIVITY PROFILES



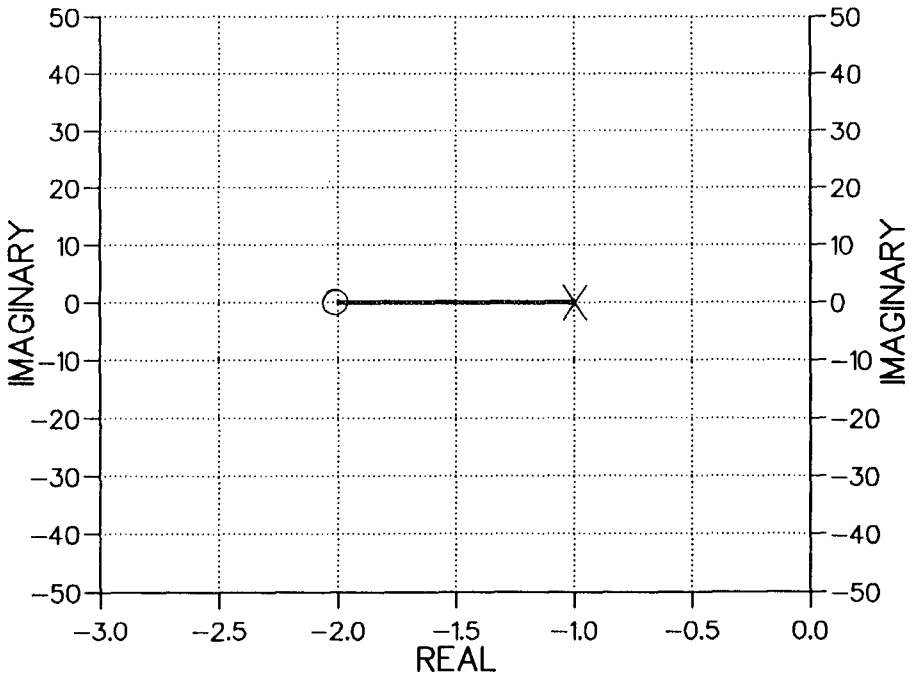
0.2 sec Delay



1.0 sec Delay

FIGURE 4.1 a FIRST-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

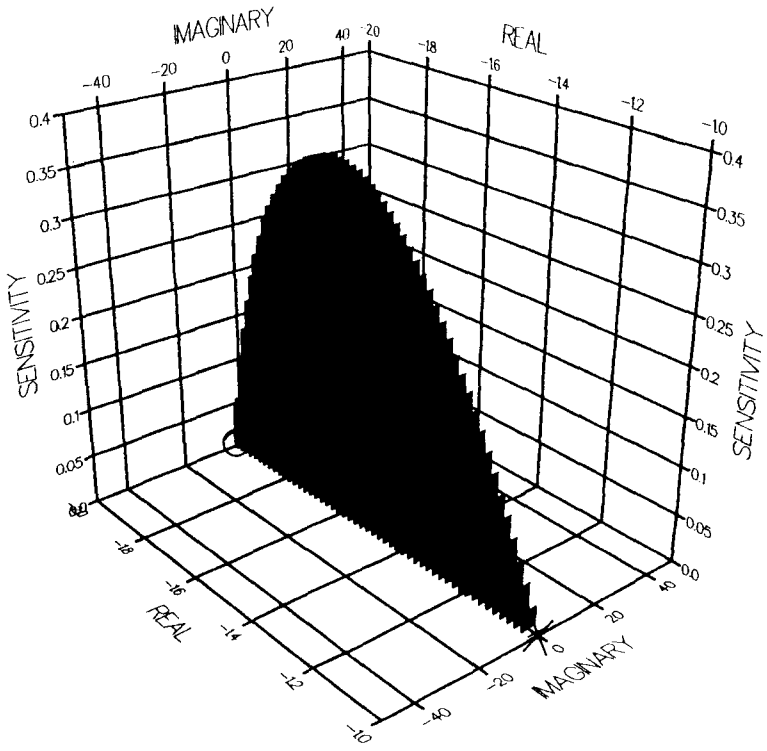
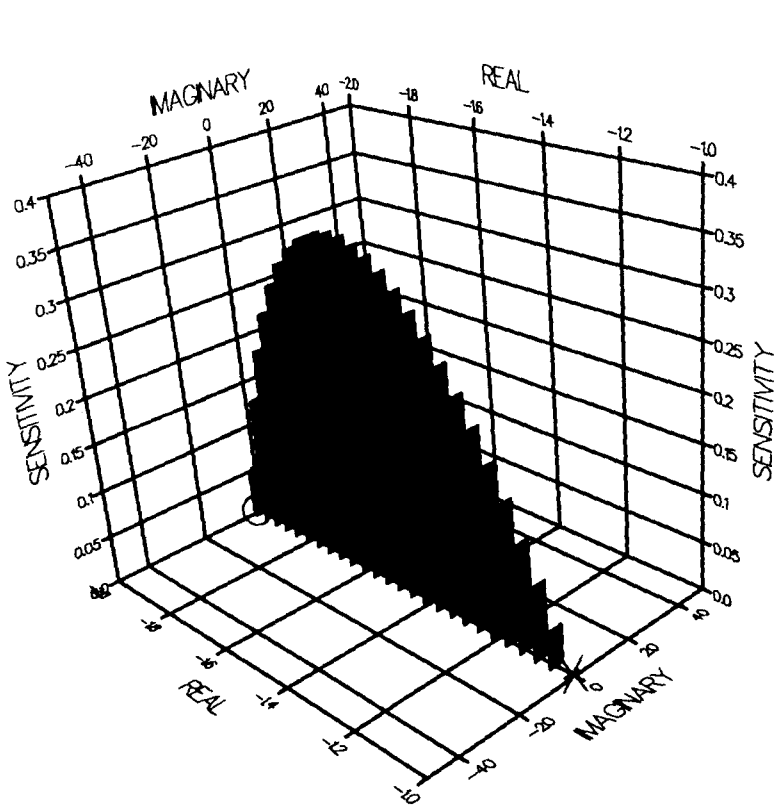
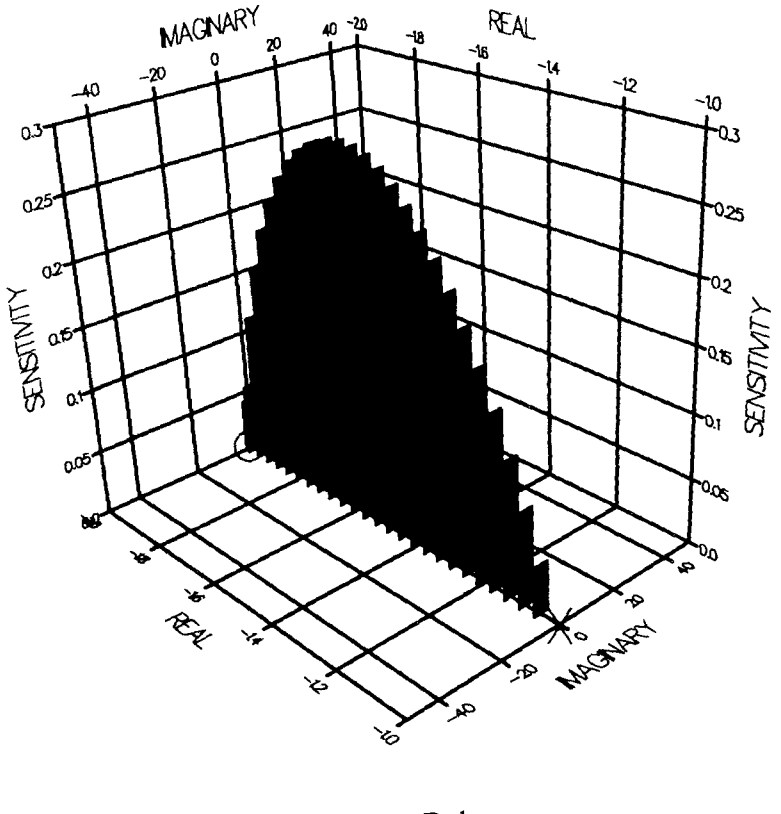


FIGURE 4.1B FIRST-ORDER PLANT

3-D SENSITIVITY PROFILES



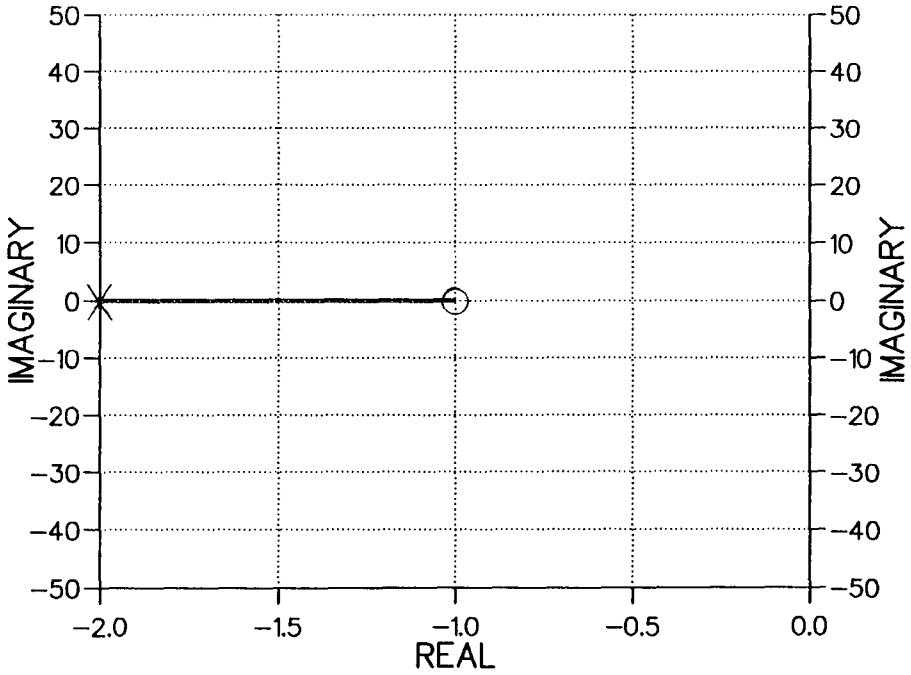
0.2 sec Delay



1.0 sec Delay

FIGURE 4.1 b FIRST-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

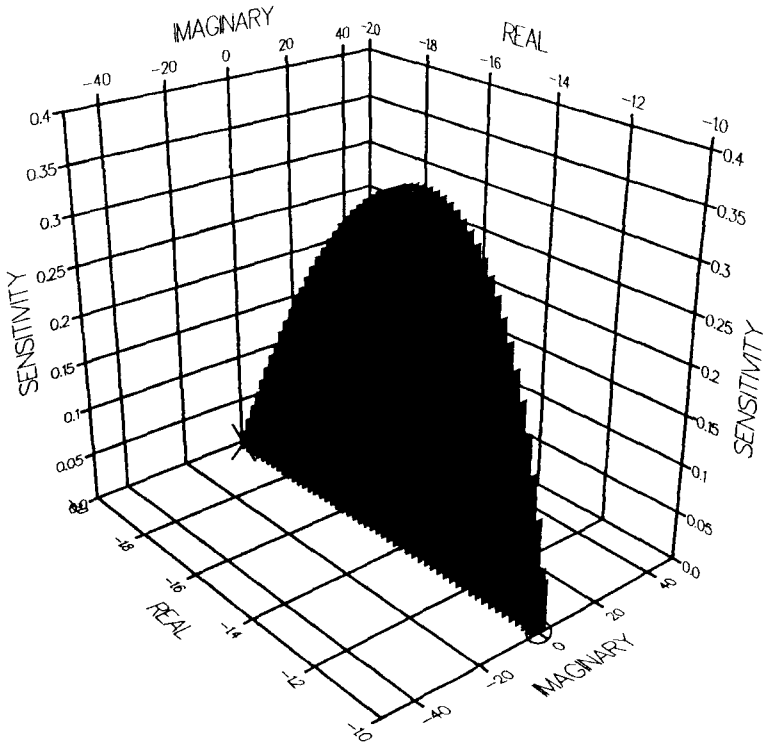
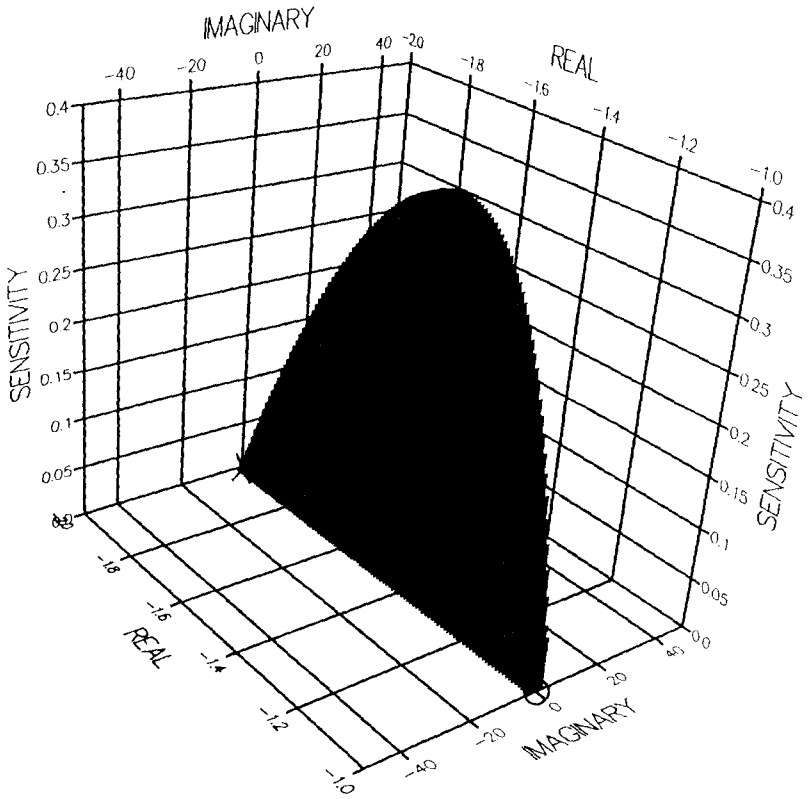
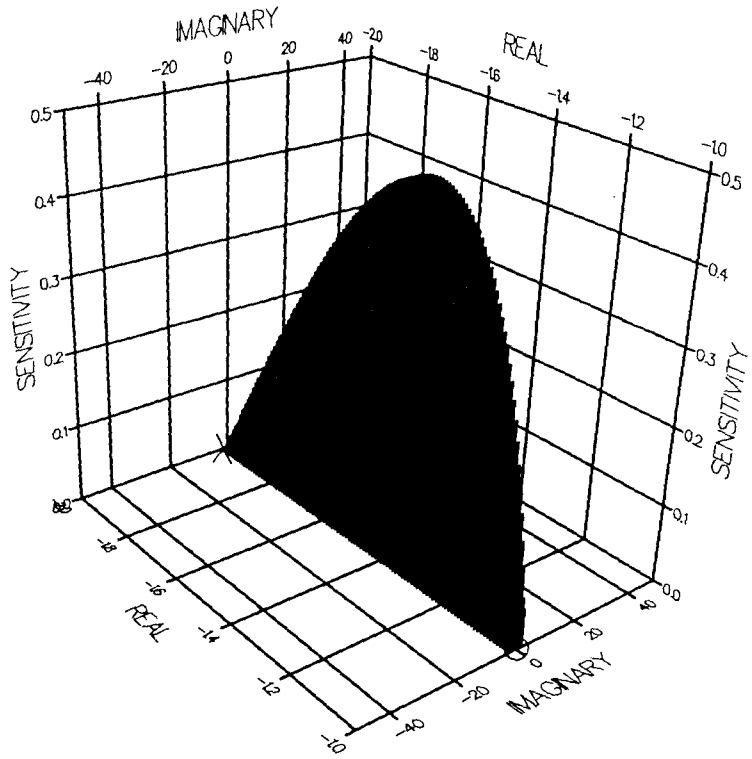


FIGURE 4.1C FIRST-ORDER PLANT

3-D SENSITIVITY PROFILES



0.2 sec Delay

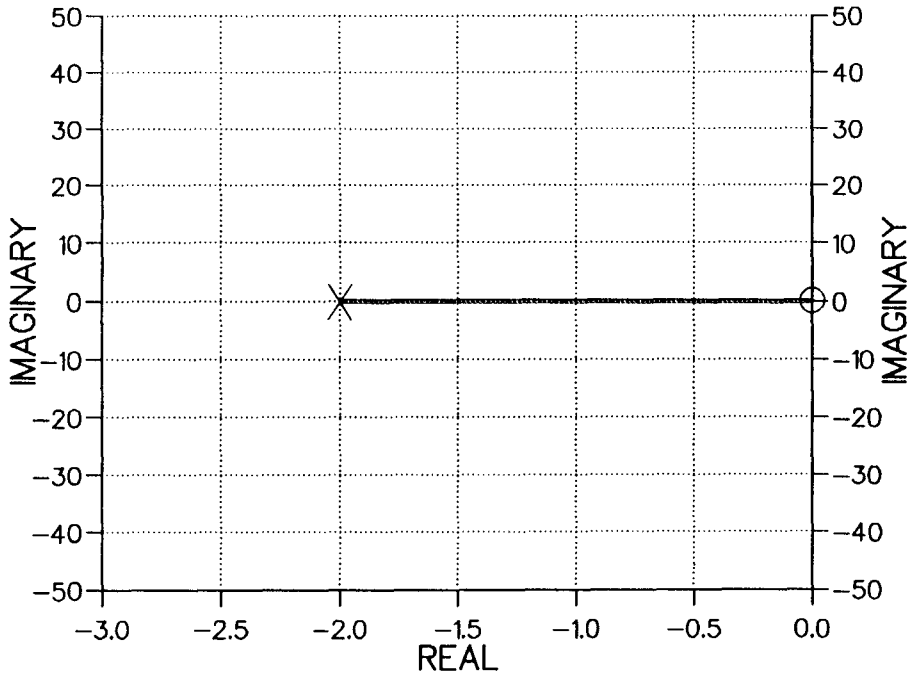


1.0 sec Delay

FIGURE 4.1 c FIRST-ORDER PLANT WITH DELAYS

ROOT LOCUS

3-D SENSITIVITY PROFILE



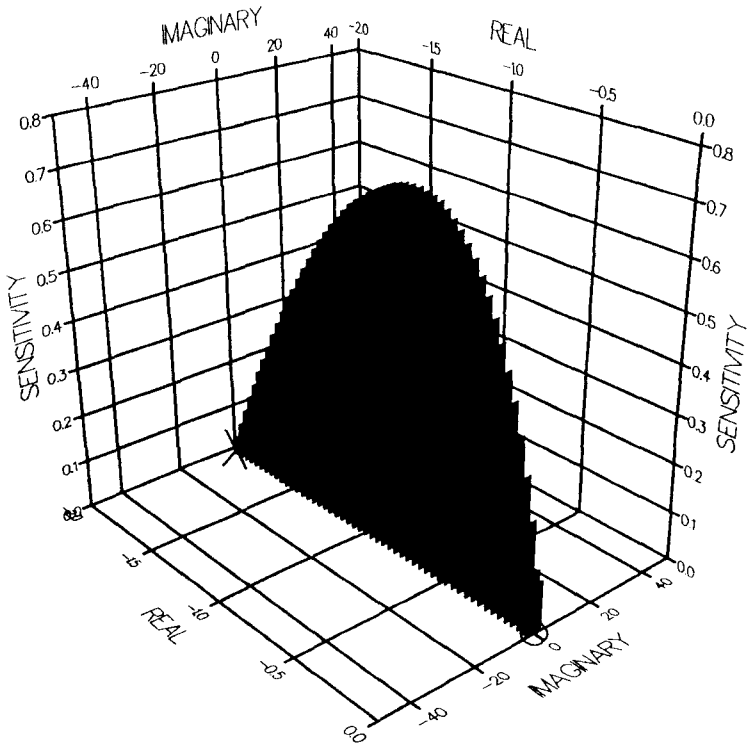
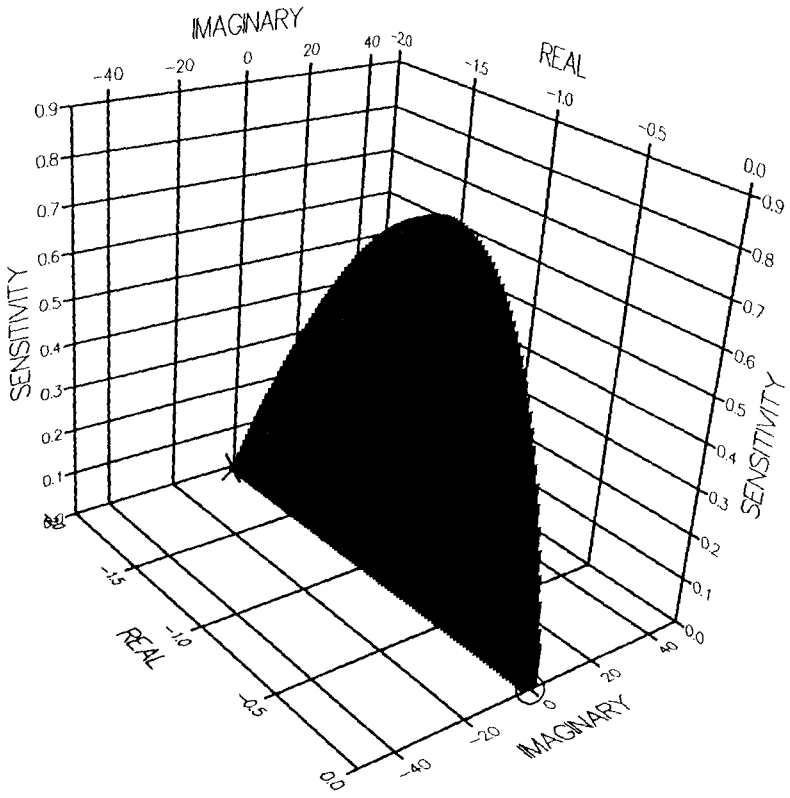
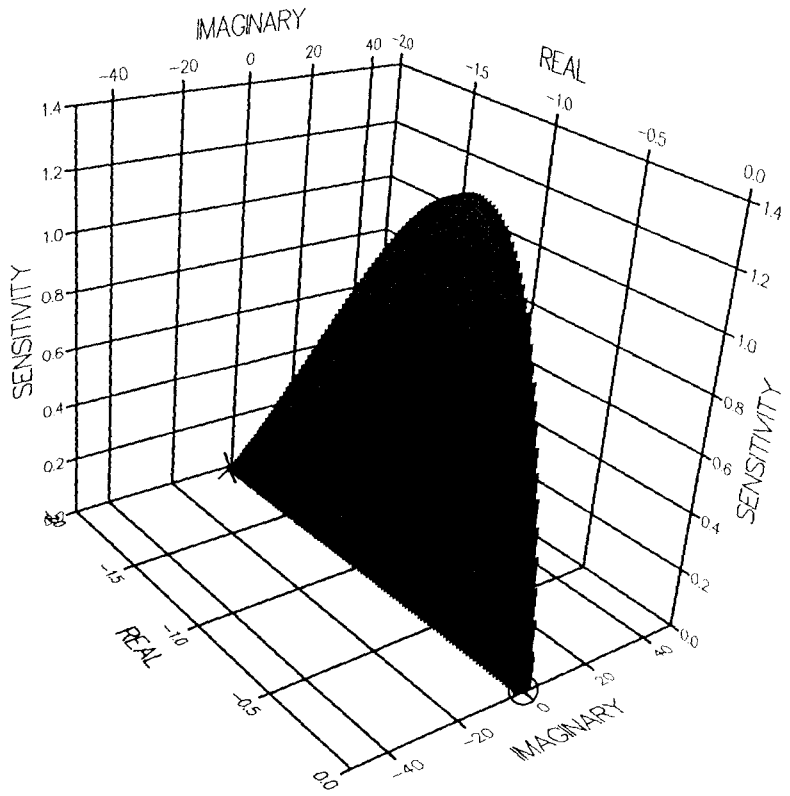


FIGURE 4.1D FIRST-ORDER PLANT

3-D SENSITIVITY PROFILES



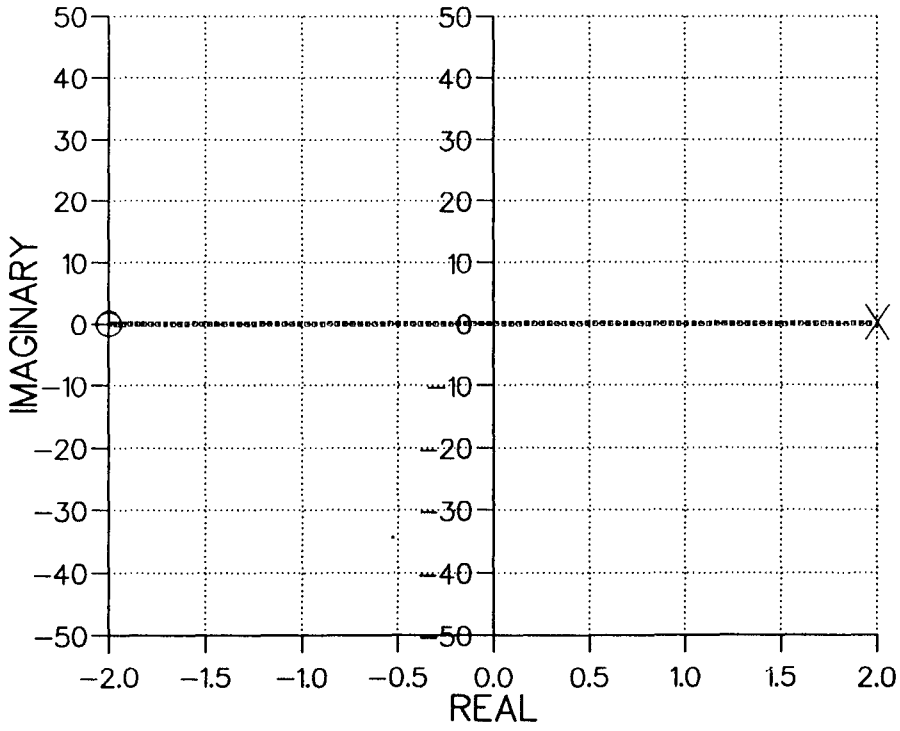
0.2 sec Delay



1.0 sec Delay

FIGURE 4.1 d FIRST-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

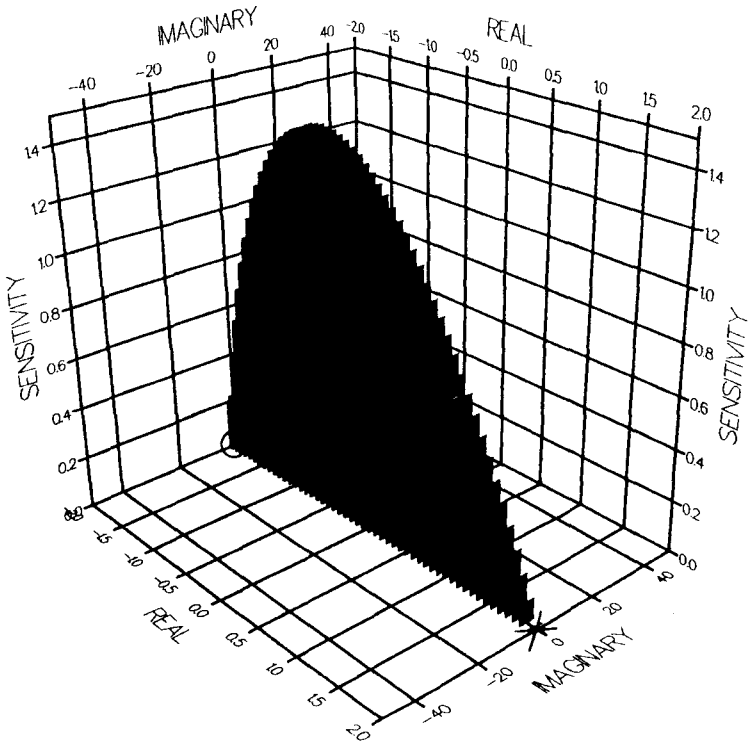
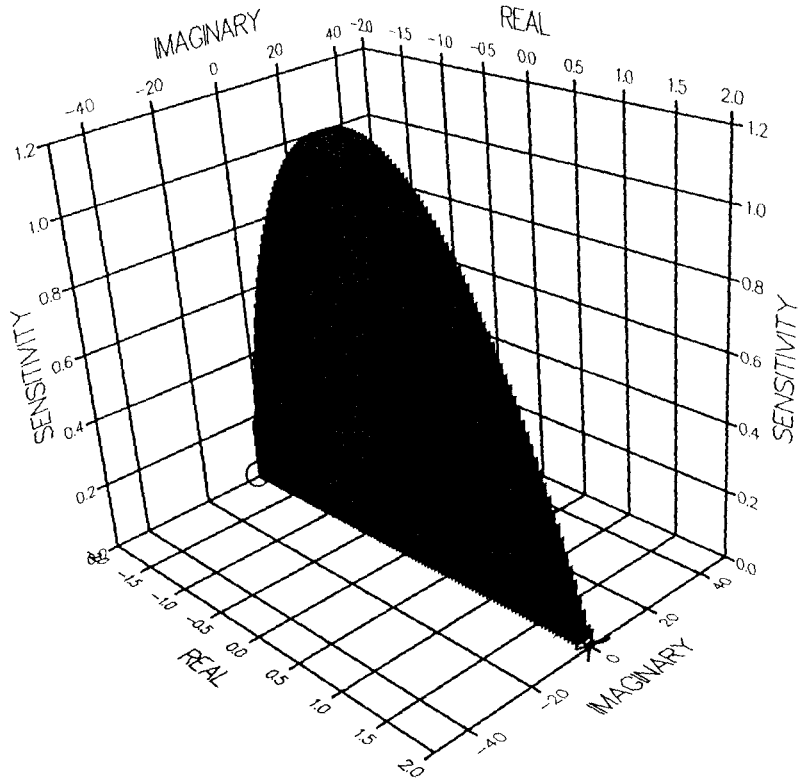
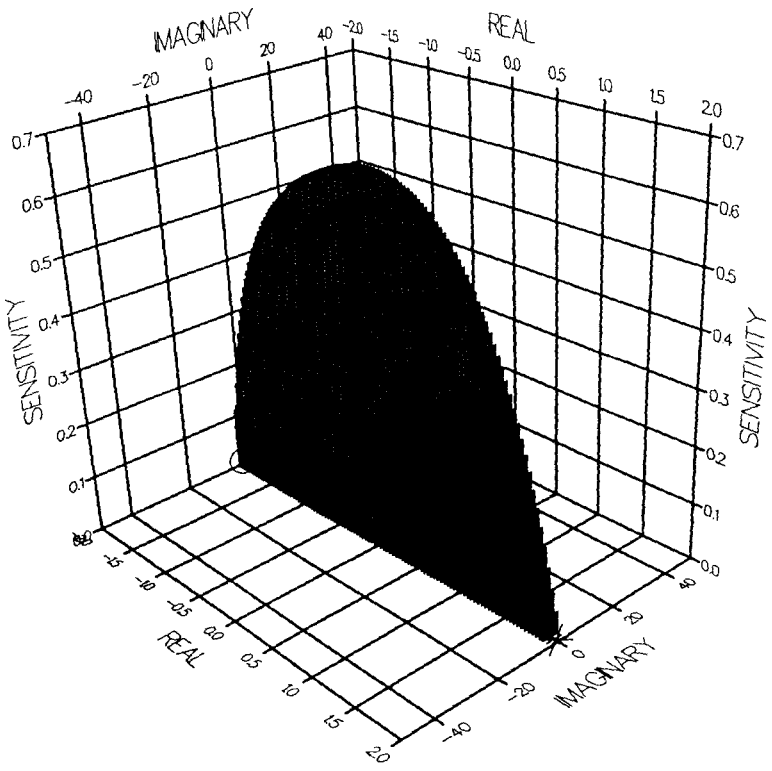


FIGURE 4.1E FIRST-ORDER PLANT

3-D SENSITIVITY PROFILES



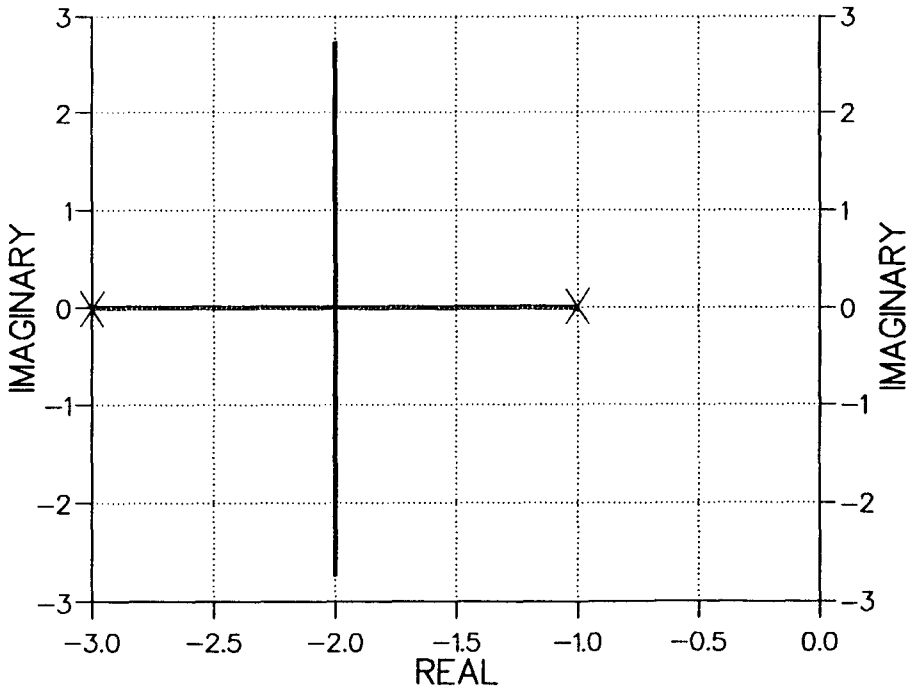
0.2 sec Delay



1.0 sec Delay

FIGURE 4.1 e FIRST-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

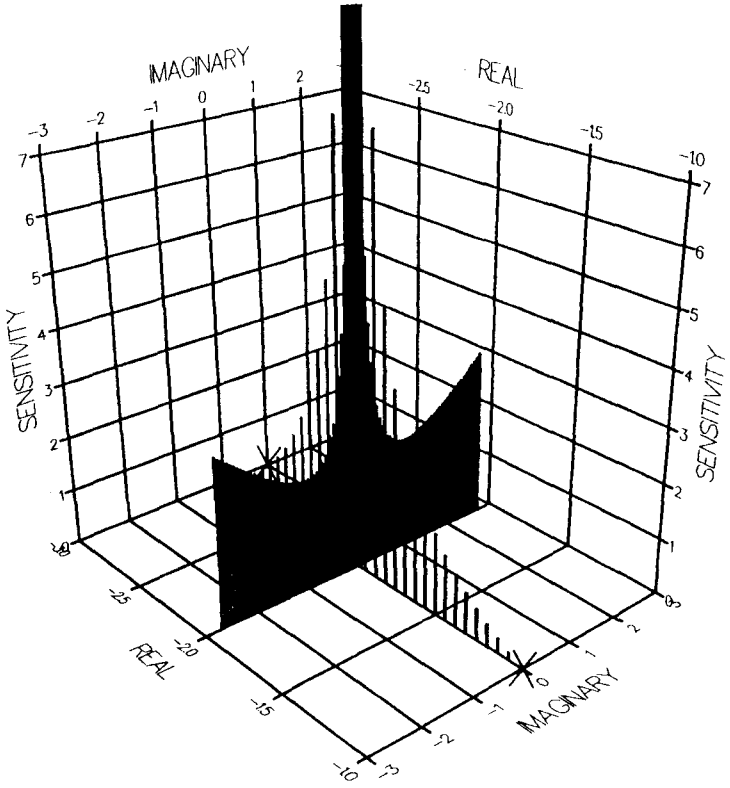
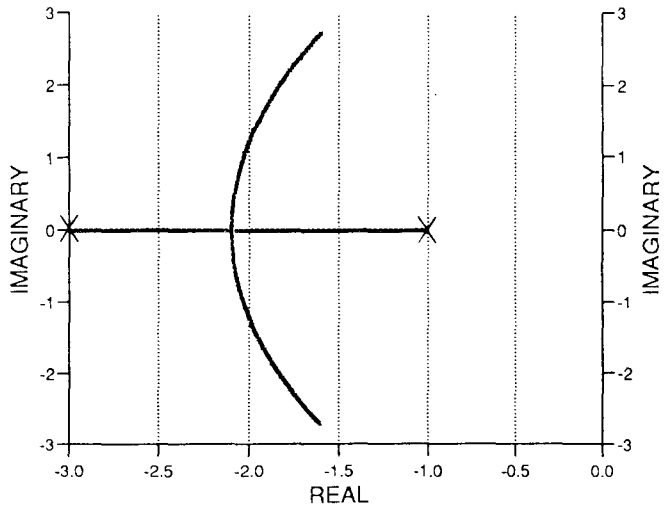
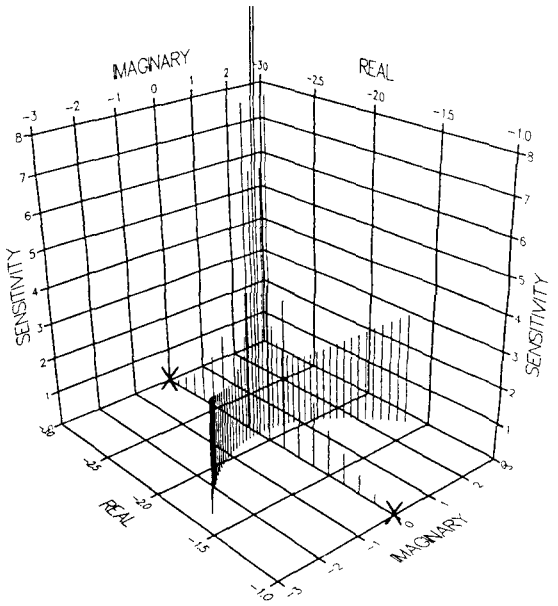


FIGURE 4.2 A SECOND-ORDER PLANT

ROOT LOCUS

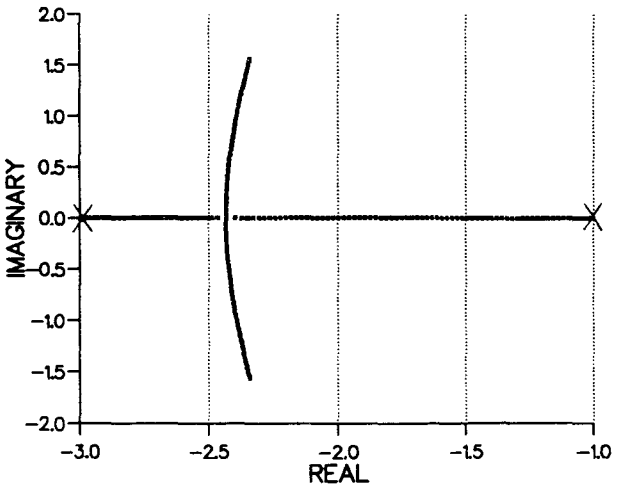


3-D SENSITIVITY PROFILE

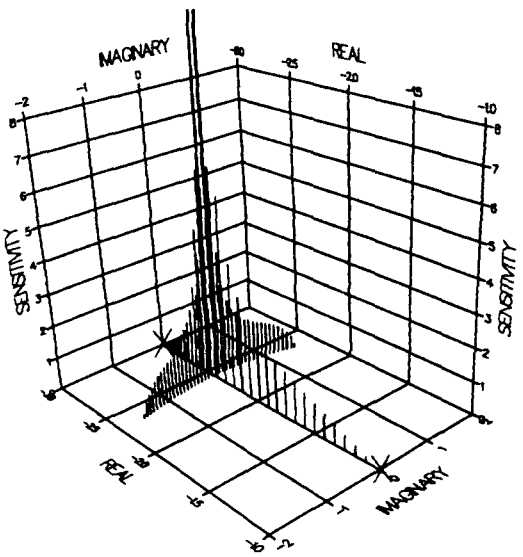


0.2 sec Delay

ROOT LOCUS



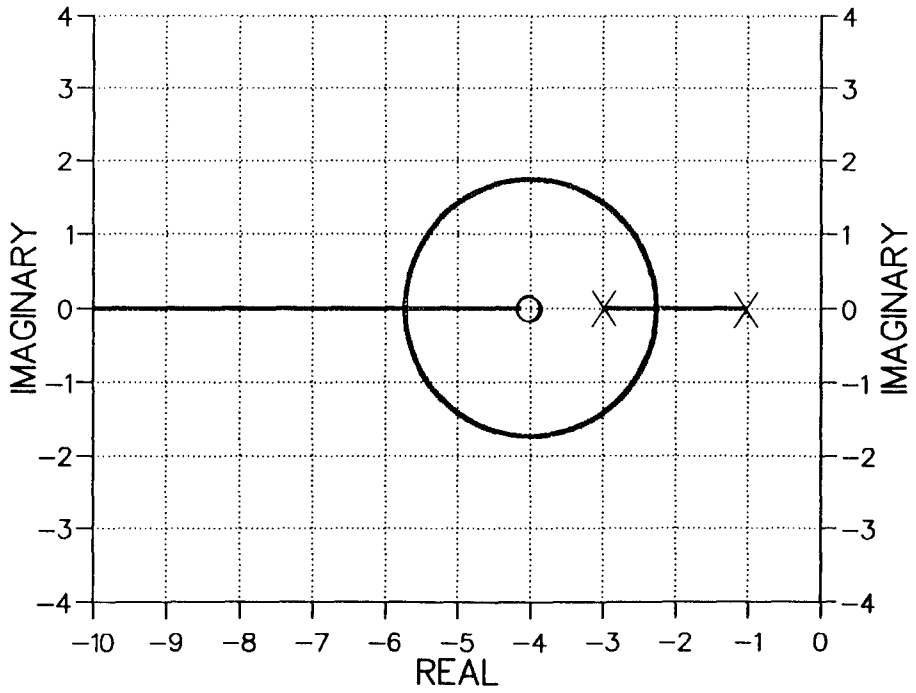
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2a SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

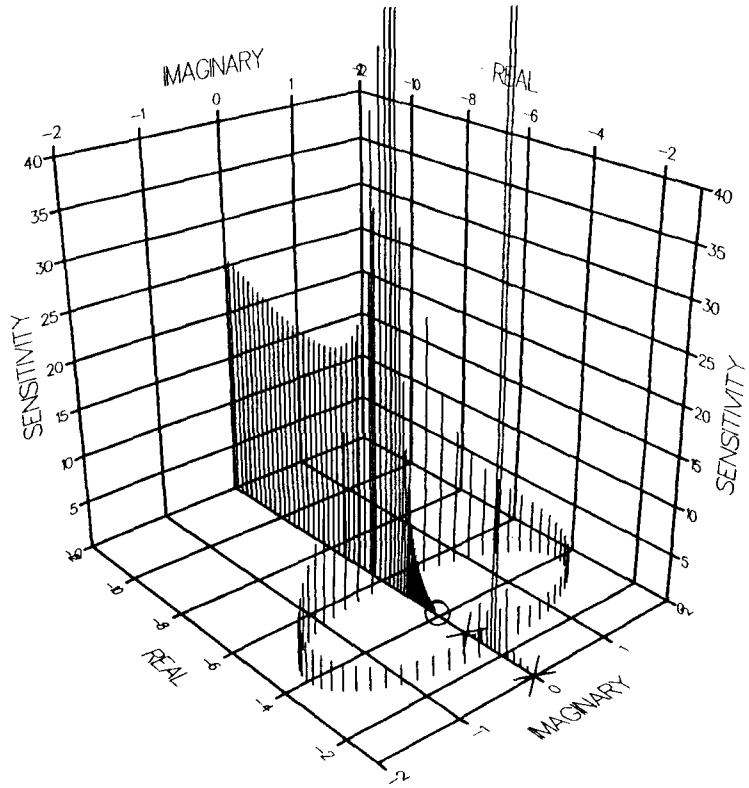
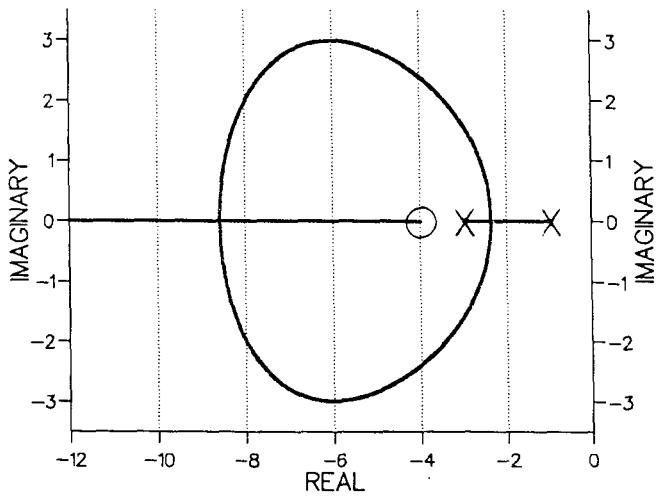
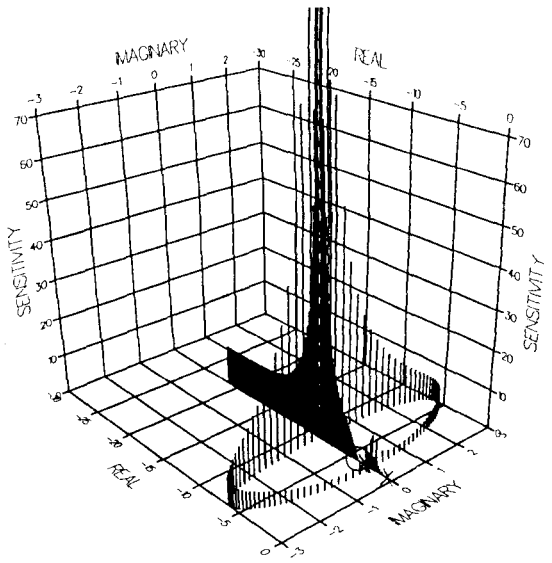


FIGURE 4.2 B SECOND-ORDER PLANT

ROOT LOCUS

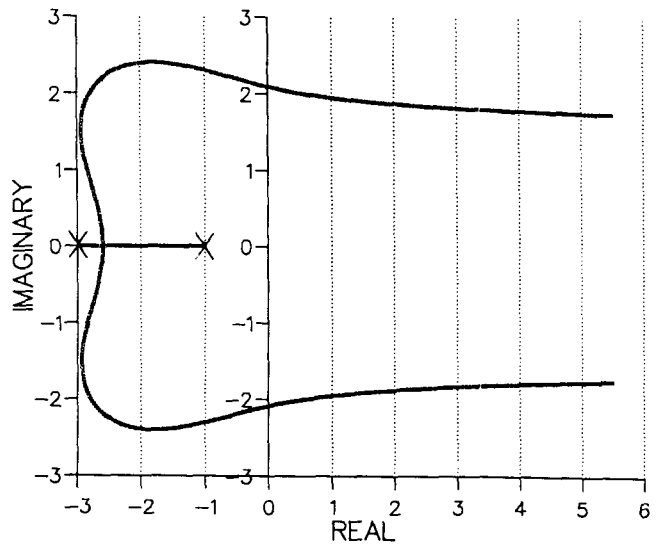


3-D SENSITIVITY PROFILE

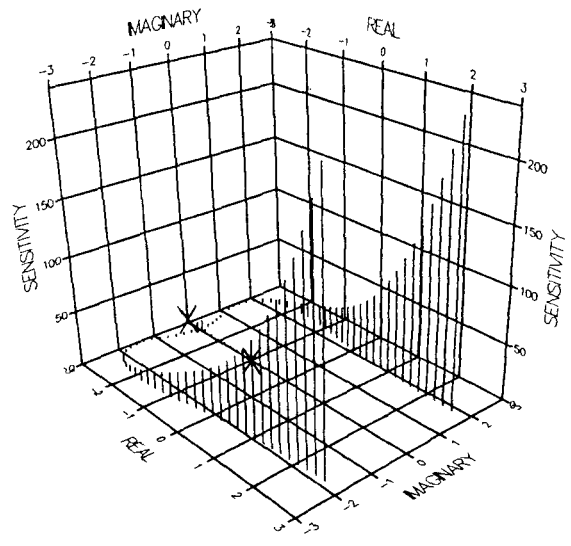


0.2 sec Delay

ROOT LOCUS



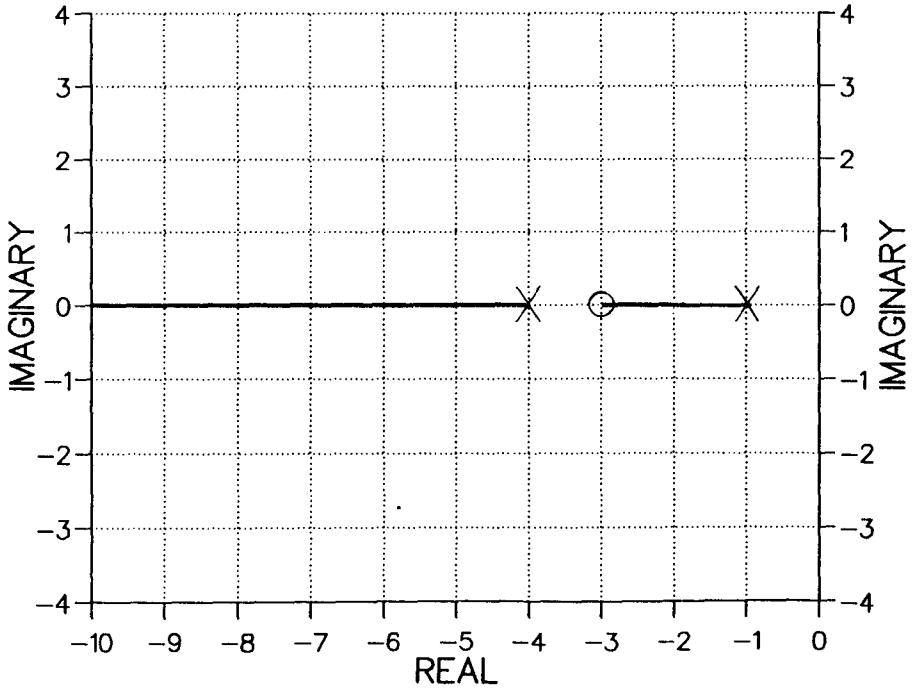
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2b SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

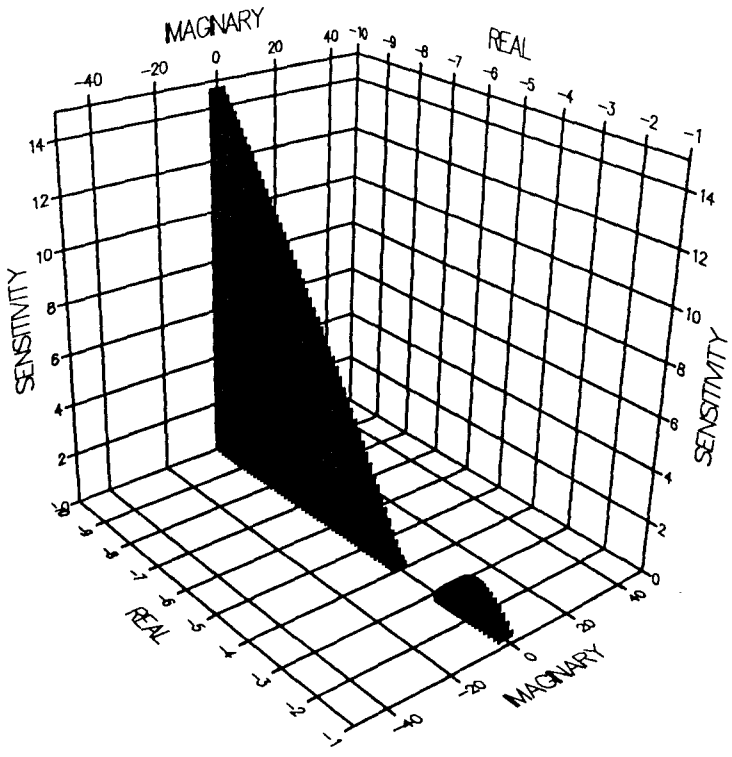
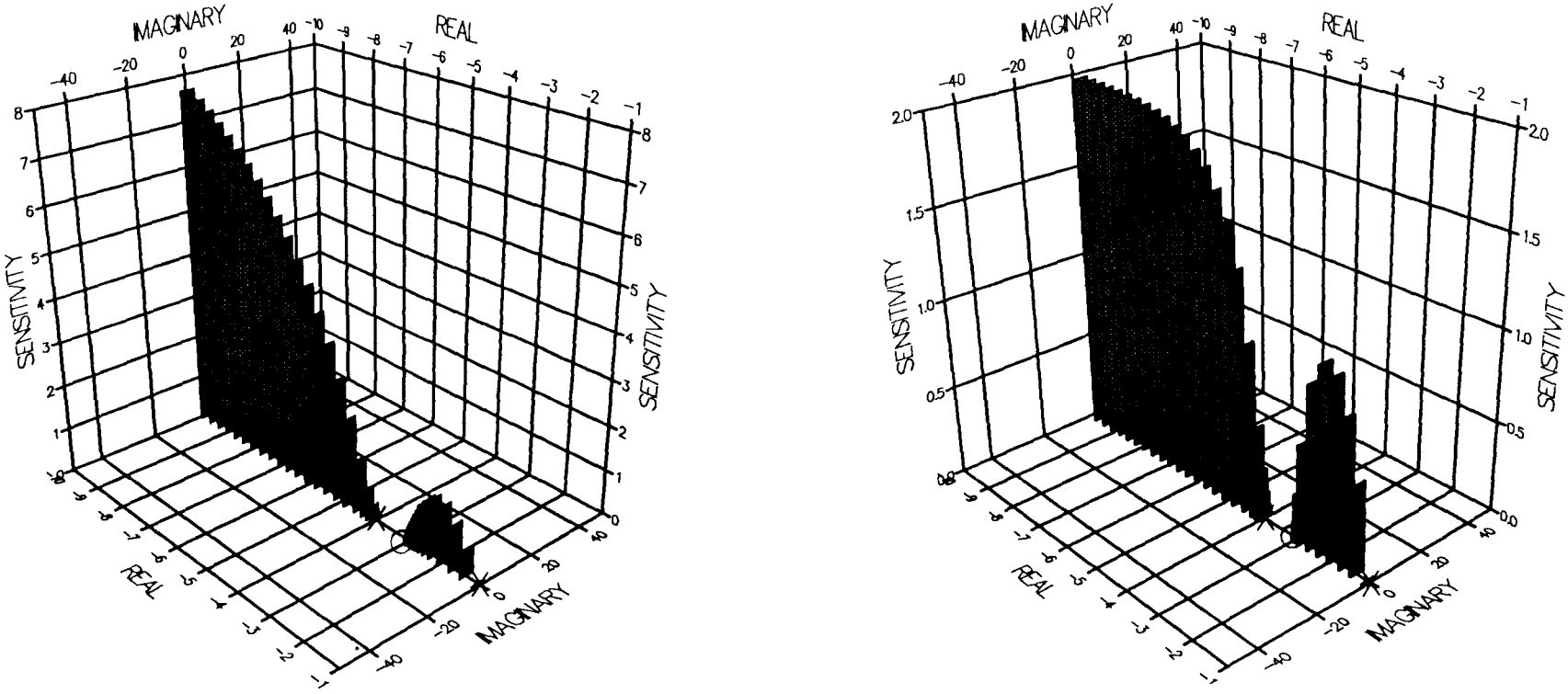


FIGURE 4.2 C SECOND-ORDER PLANT

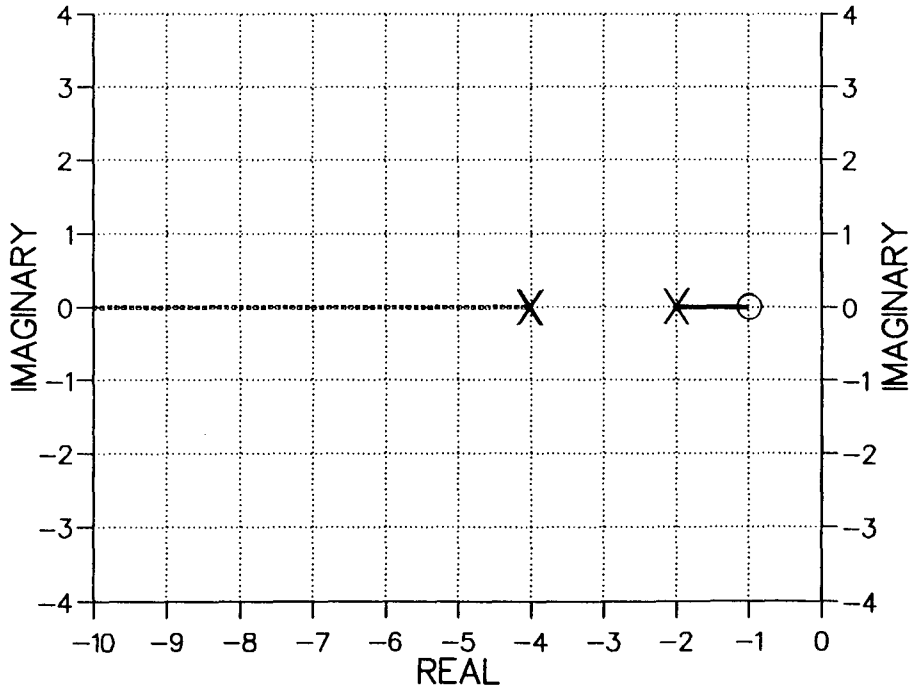


0.2 sec Delay

1.0 sec Delay

FIGURE 4.2c SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

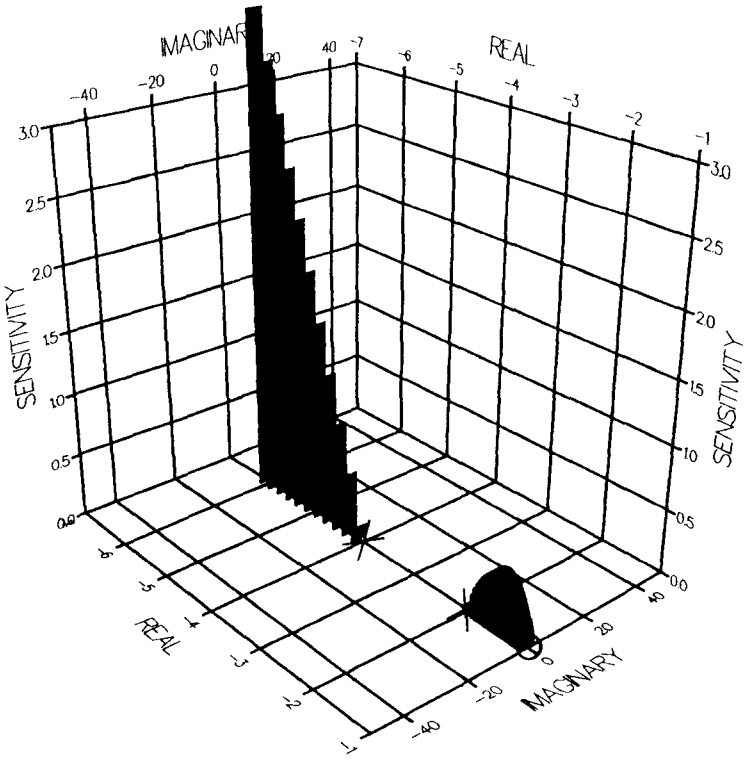
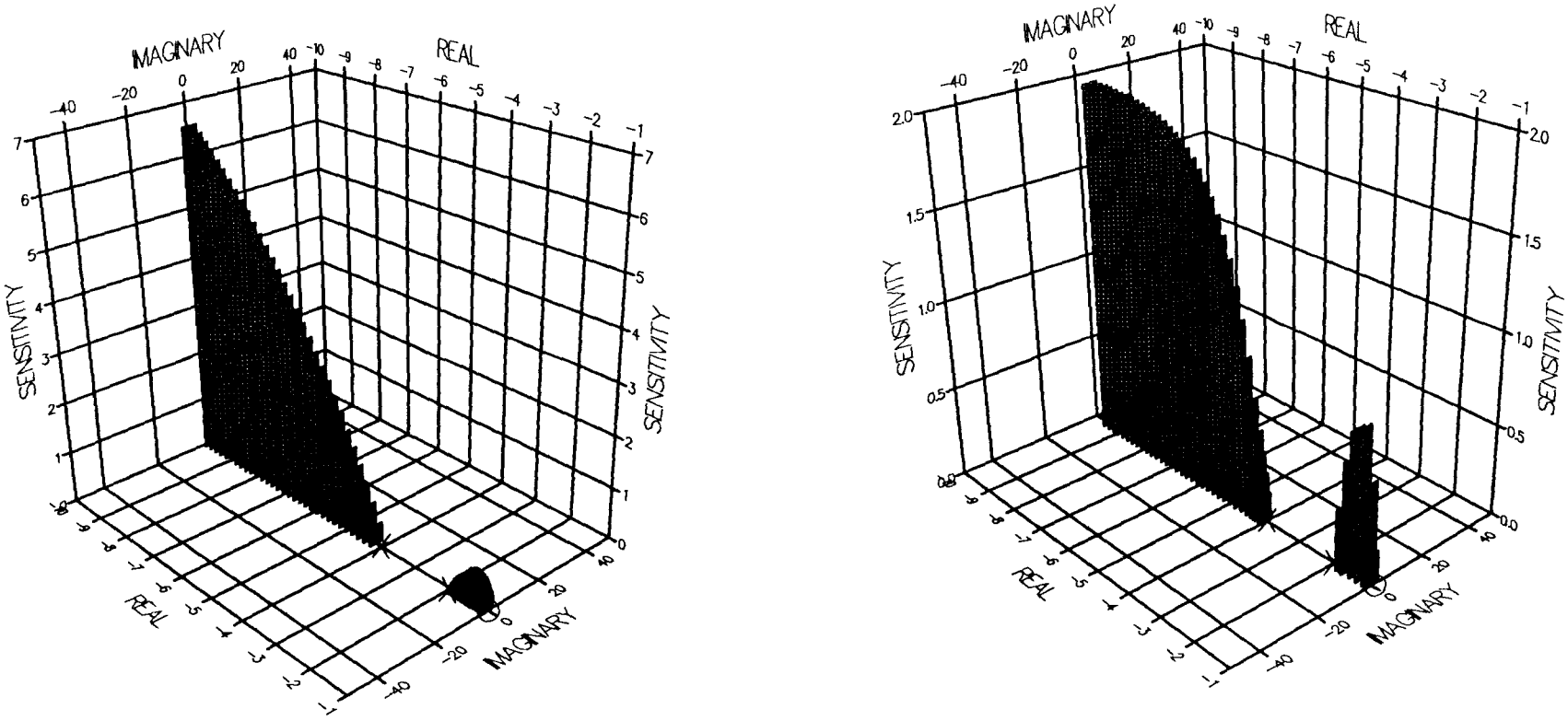


FIGURE 4.2 D SECOND-ORDER PLANT

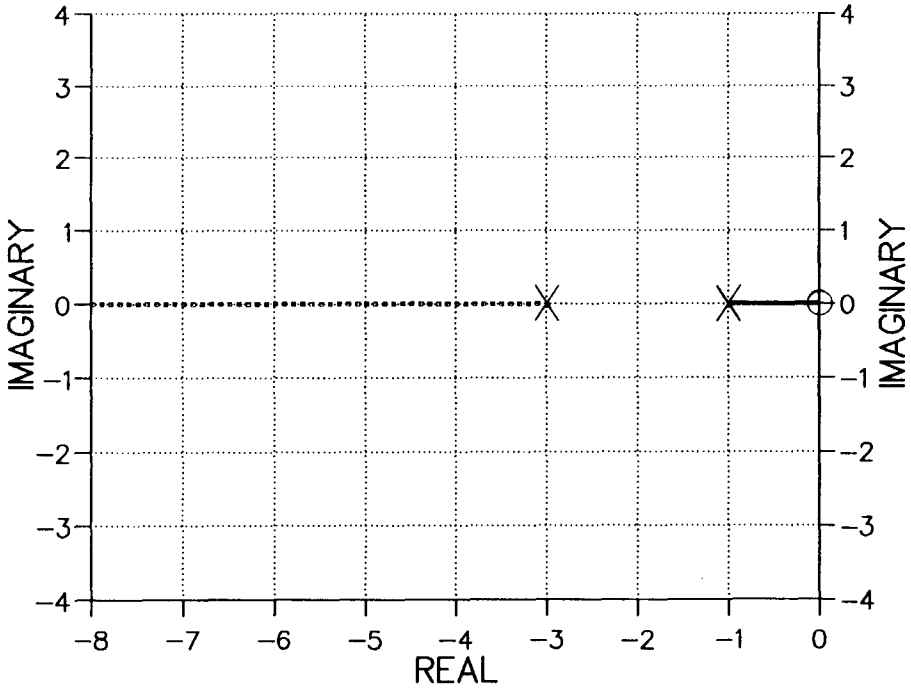


0.2 sec Delay

1.0 sec Delay

FIGURE 4.2d SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

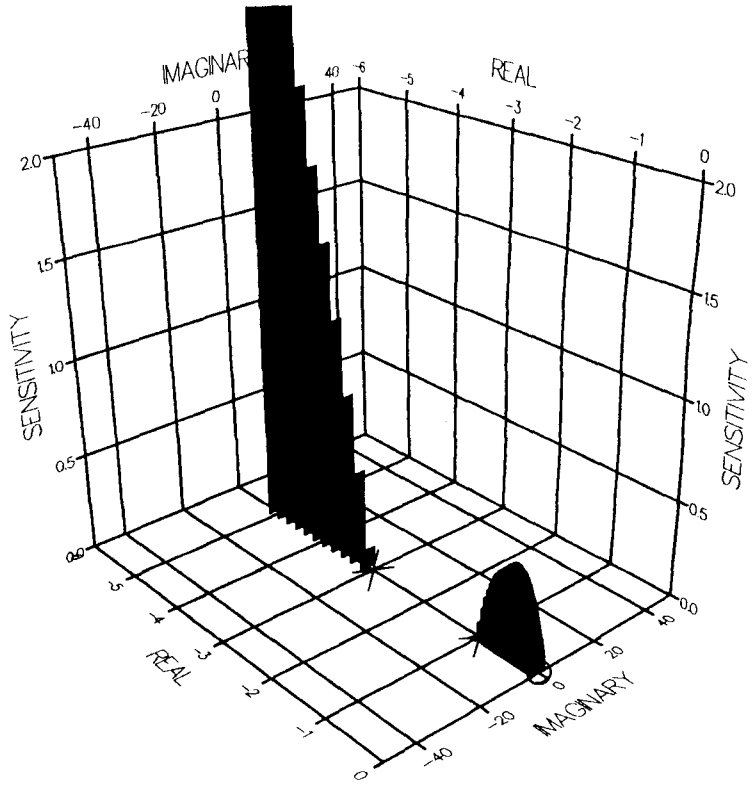
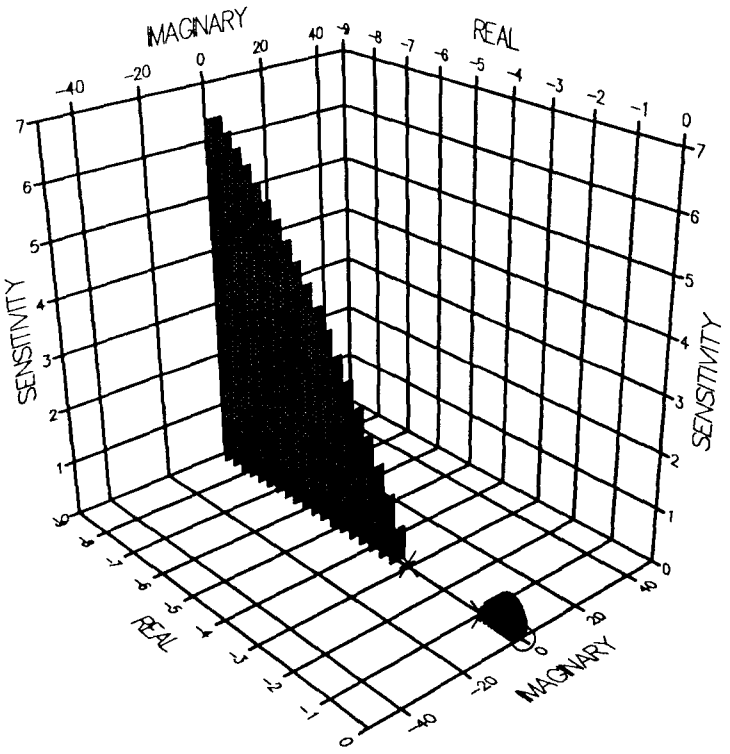
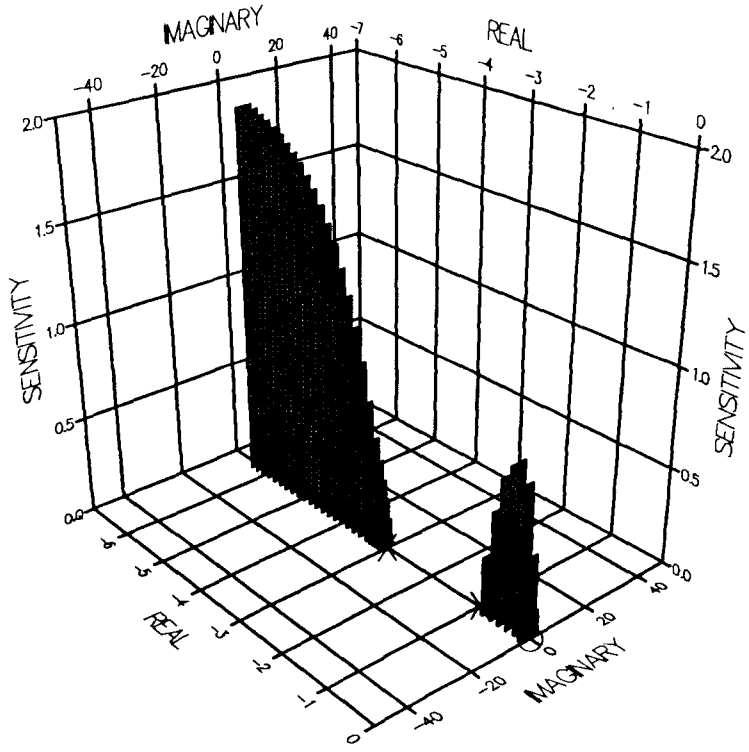


FIGURE 4.2 E SECOND-ORDER PLANT



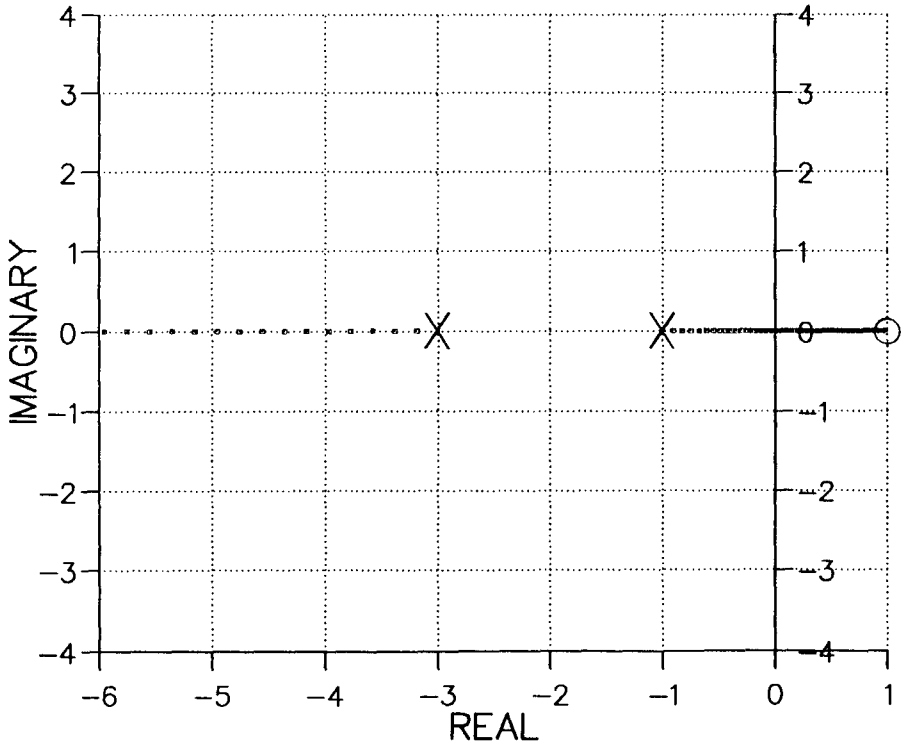
0.2 sec Delay



1.0 sec Delay

FIGURE 4.2e SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

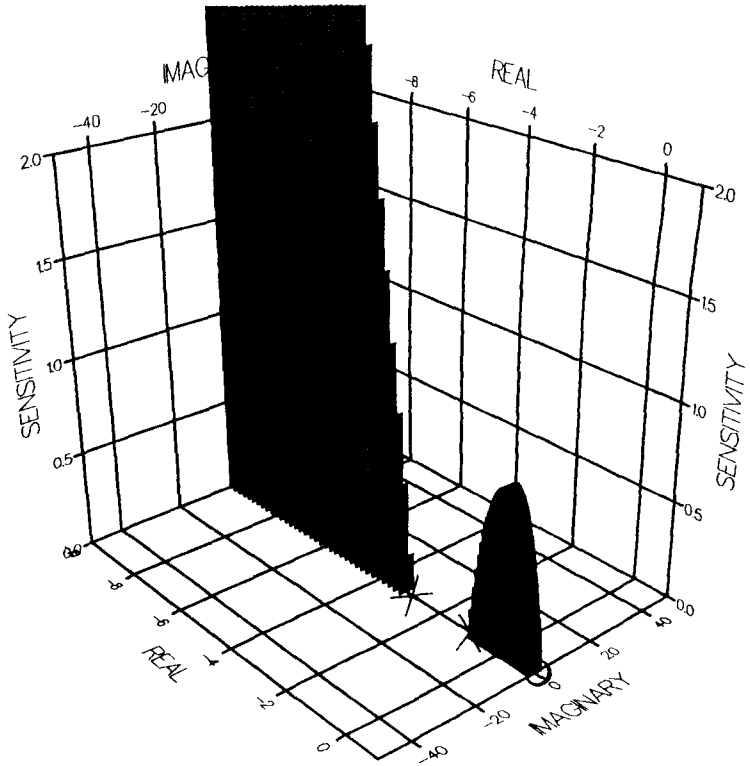
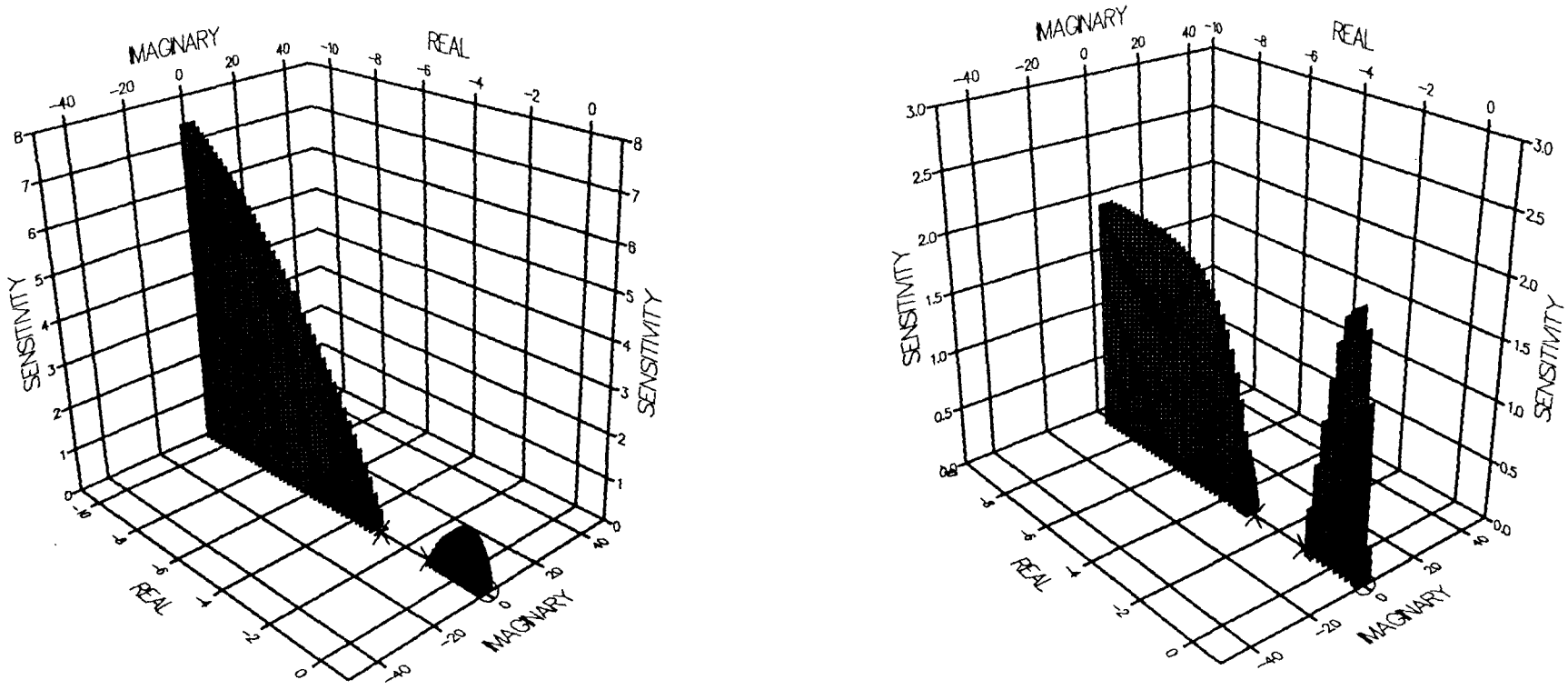


FIGURE 4.2 F SECOND-ORDER PLANT

3-D SENSITIVITY PROFILES

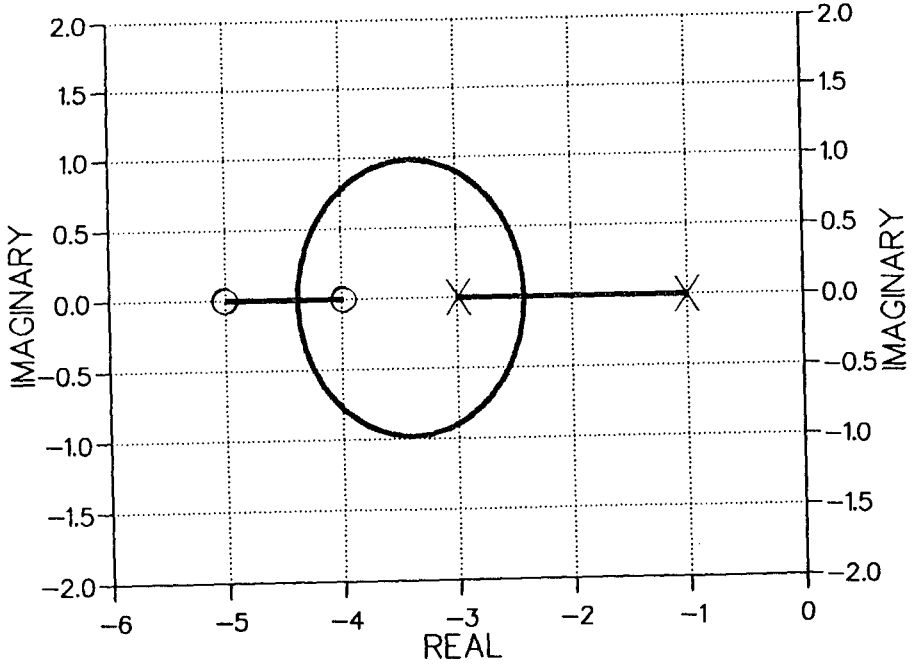


0.2 sec Delay

1.0 sec Delay

FIGURE 4.2f SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

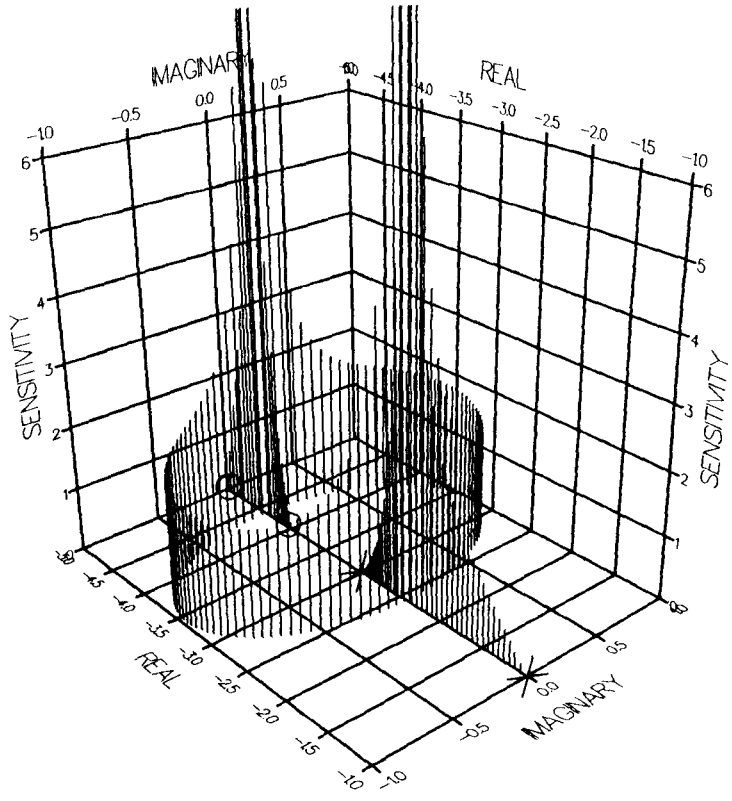
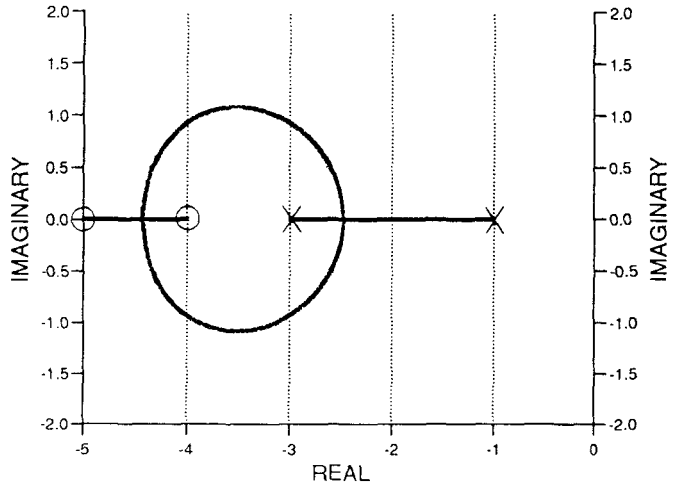
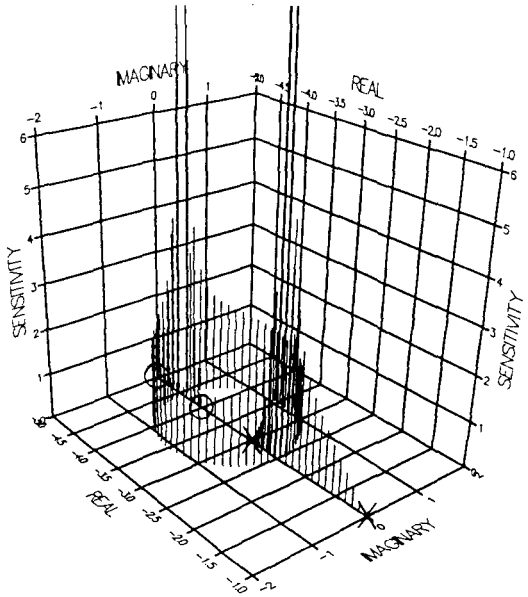


FIGURE 4.2 G SECOND-ORDER PLANT

ROOT LOCUS

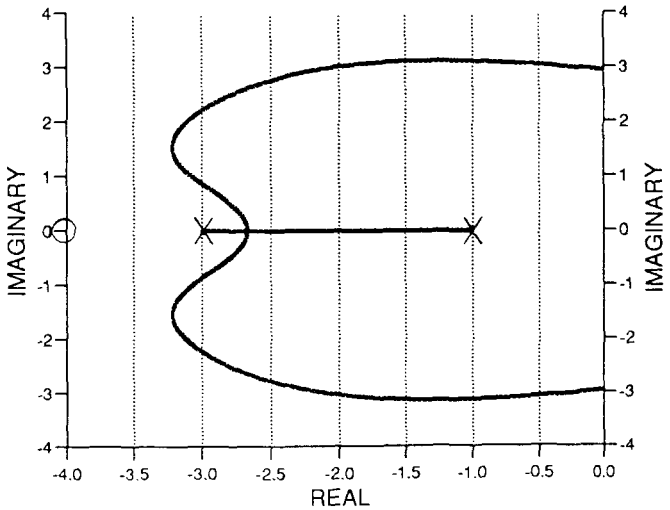


3-D SENSITIVITY PROFILE

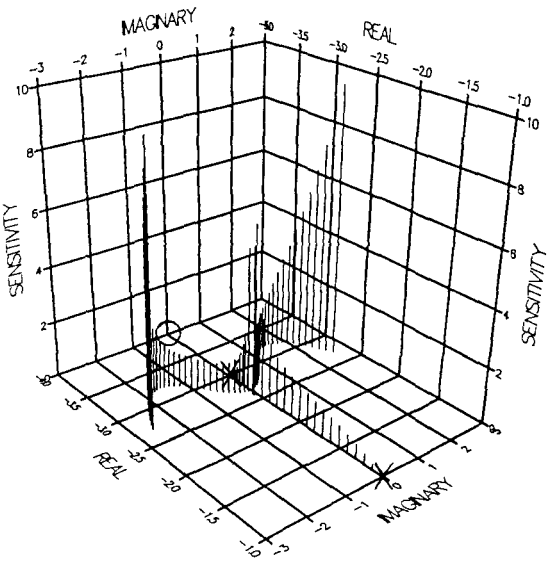


0.2 sec Delay

ROOT LOCUS



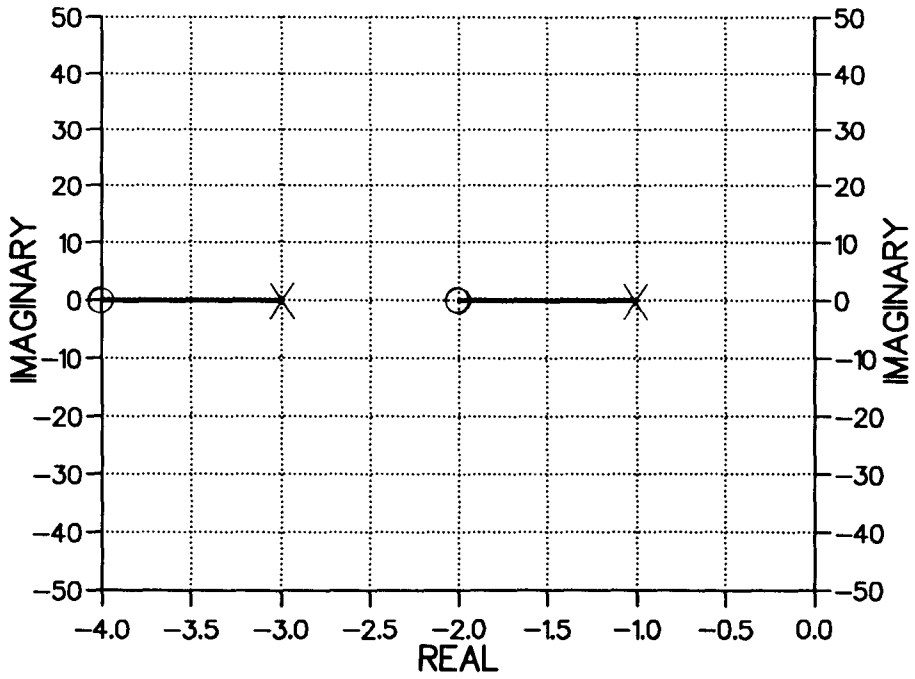
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 g SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

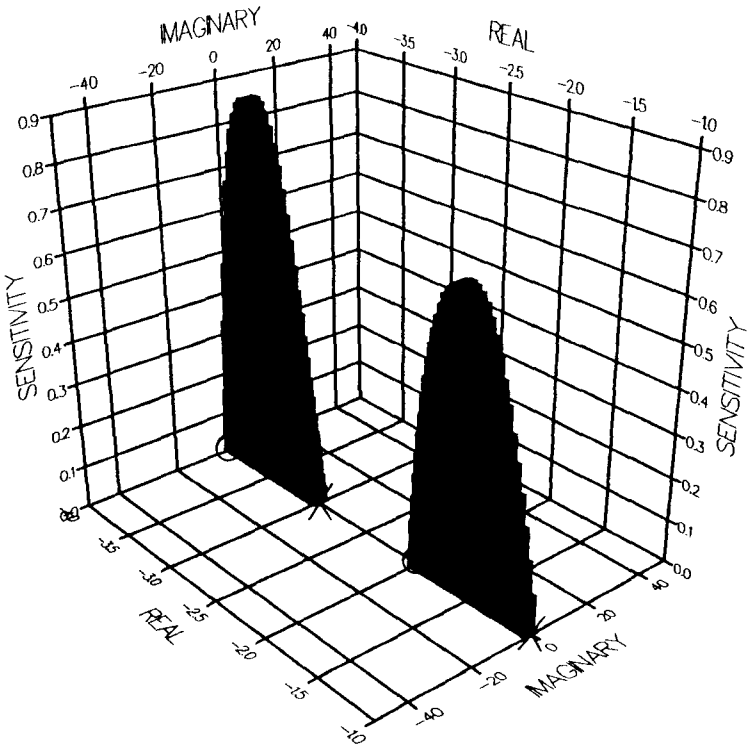
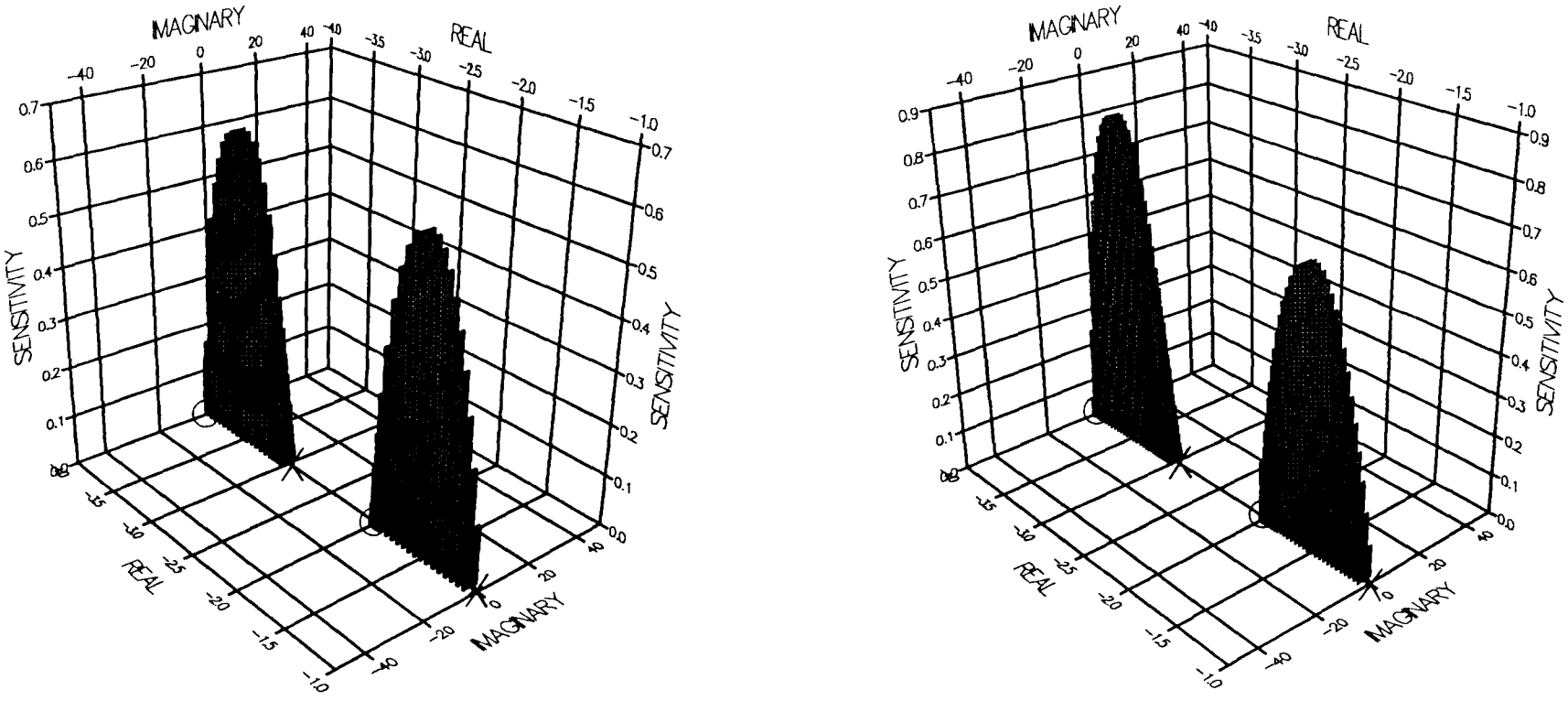


FIGURE 4.2 H SECOND-ORDER PLANT

3-D SENSITIVITY PROFILES

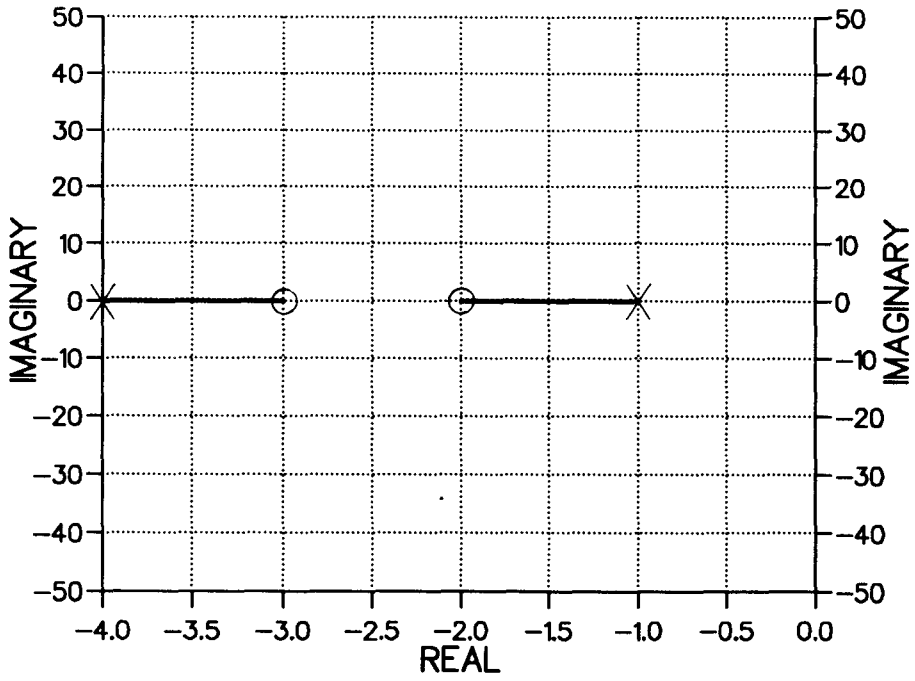


0.2 sec Delay

1.0 sec Delay

FIGURE 4.2 h SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

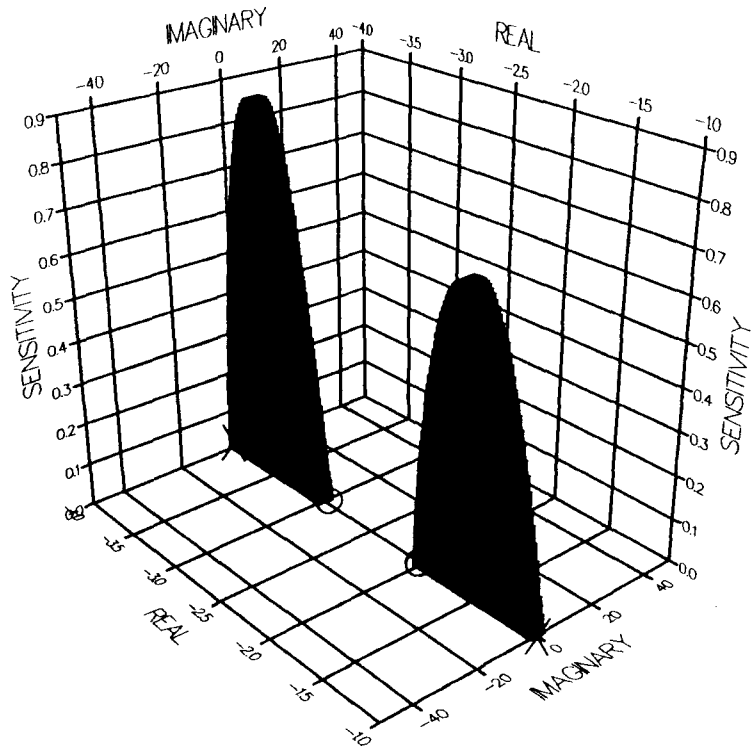
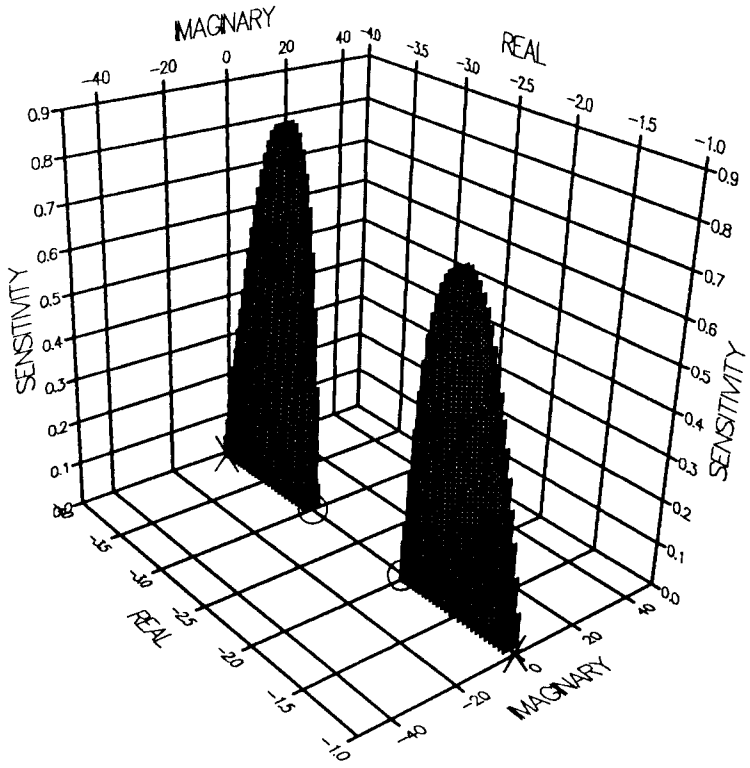
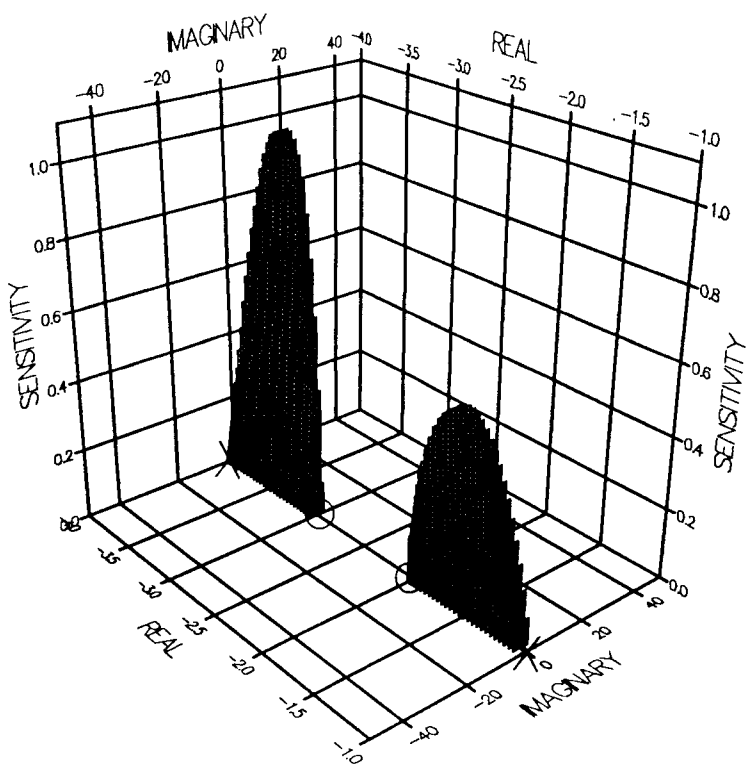


FIGURE 4.2 I SECOND-ORDER PLANT

3-D SENSITIVITY PROFILES



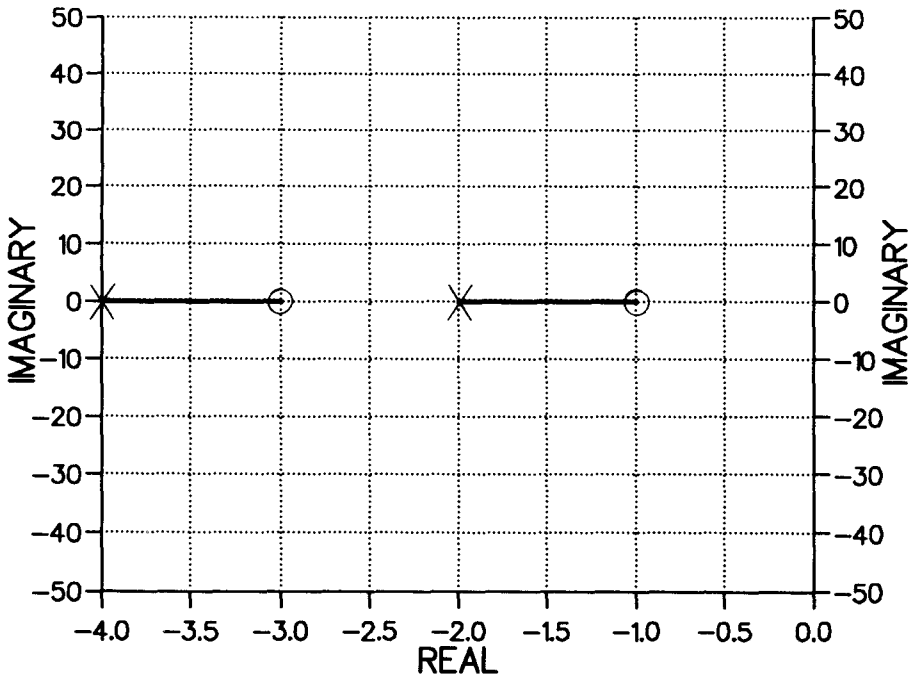
0.2 sec Delay



1.0 sec Delay

FIGURE 4.2 i SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

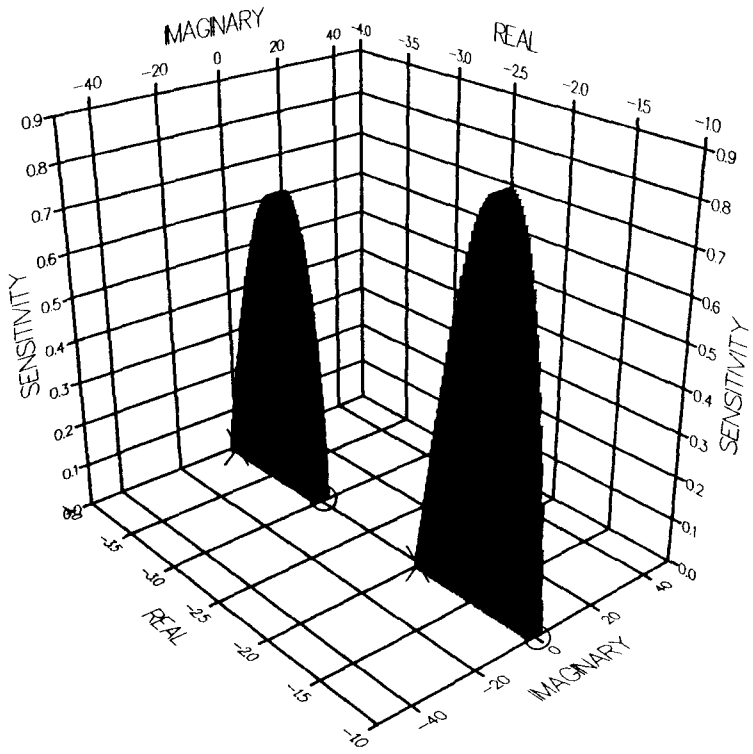
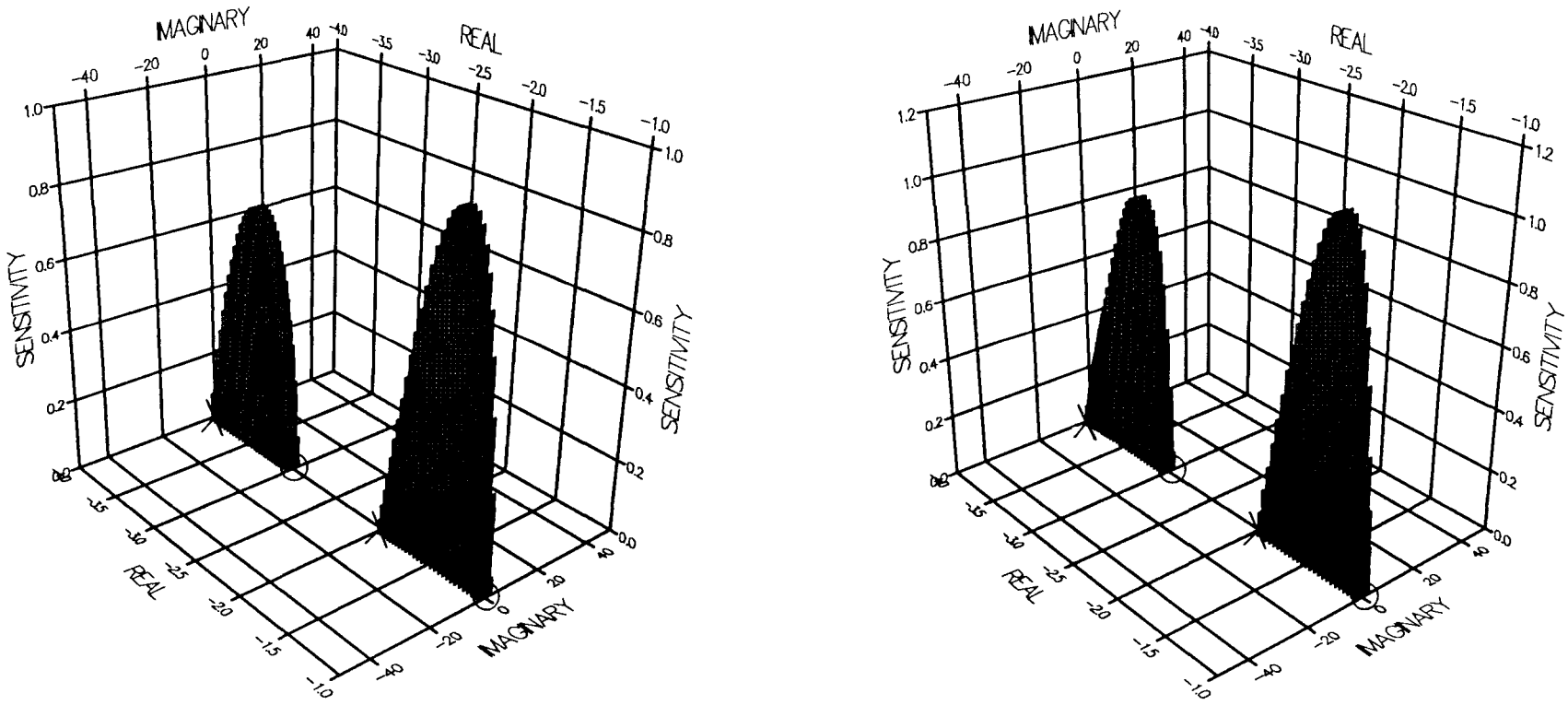


FIGURE 4.2 J SECOND-ORDER PLANT

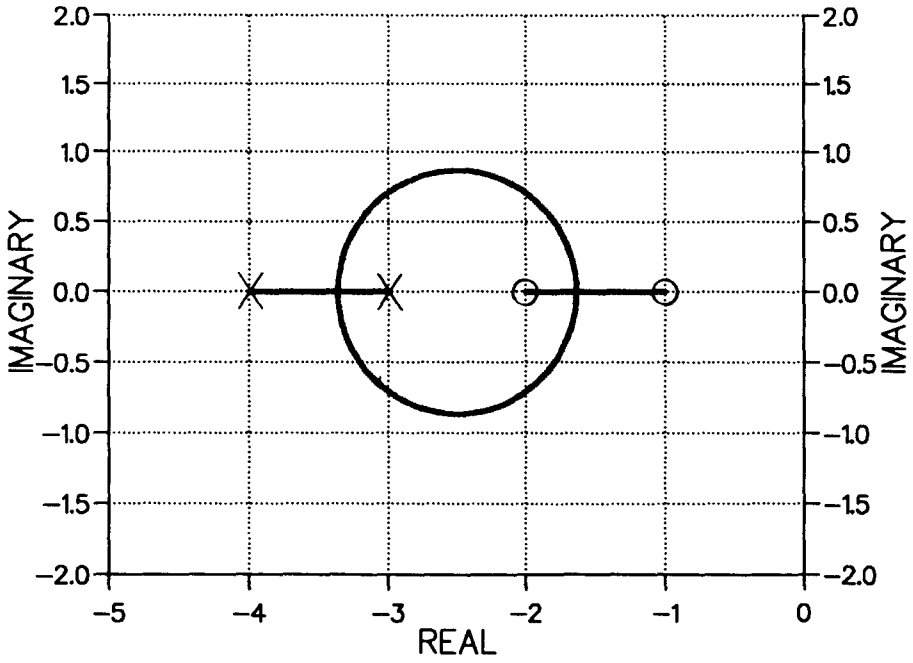


0.2 sec Delay

1.0 sec Delay

FIGURE 4.2 j SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

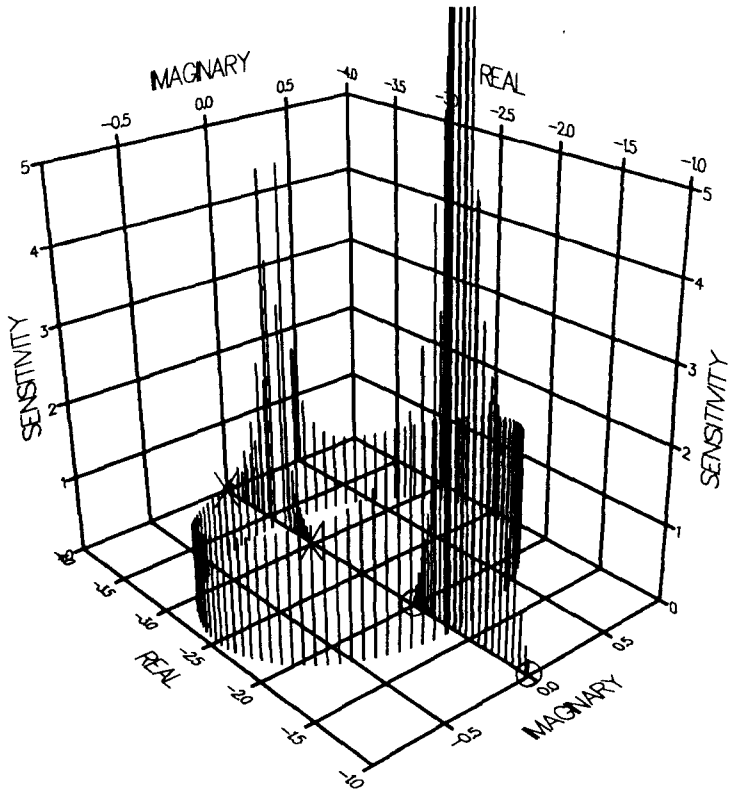
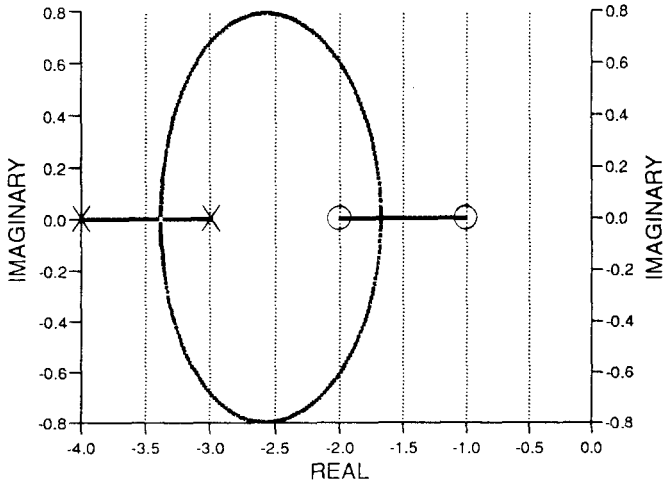
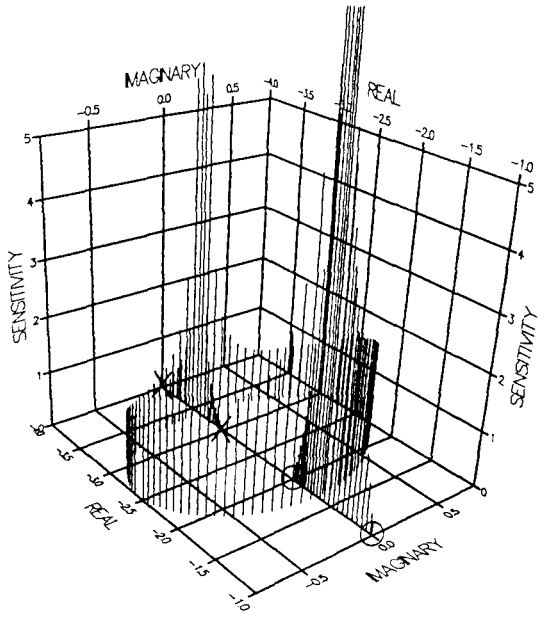


FIGURE 4.2 K SECOND-ORDER PLANT

ROOT LOCUS

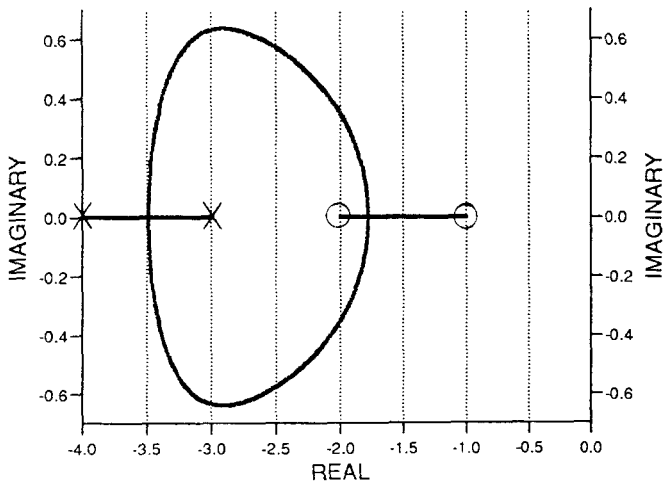


3-D SENSITIVITY PROFILE

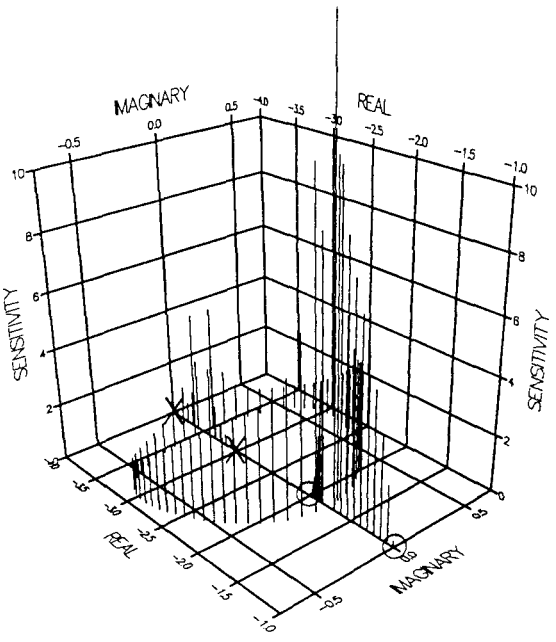


0.2 sec Delay

ROOT LOCUS



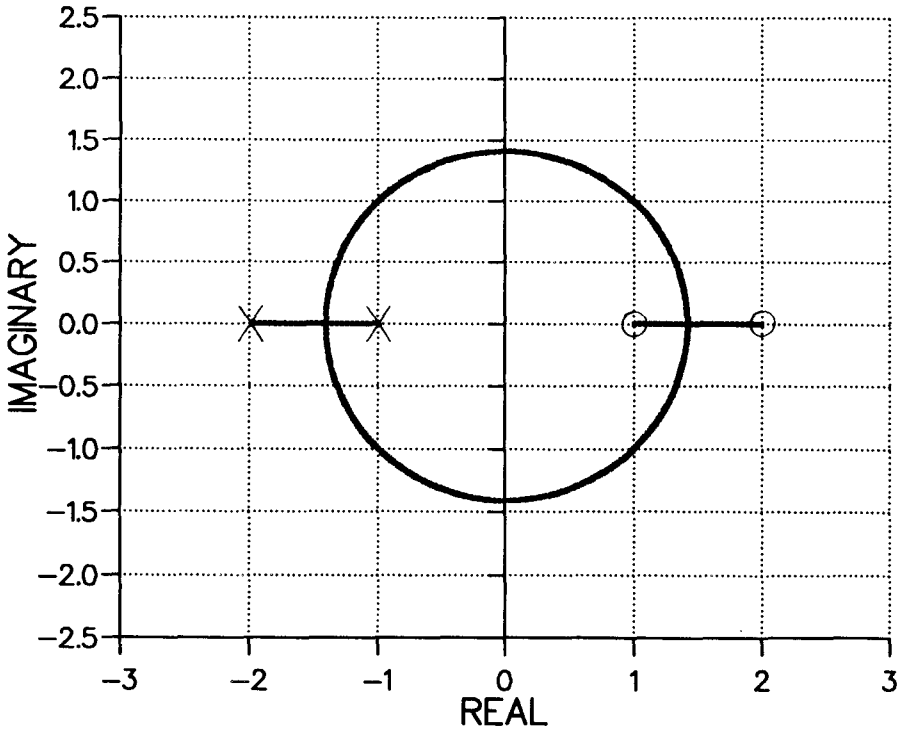
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 k SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

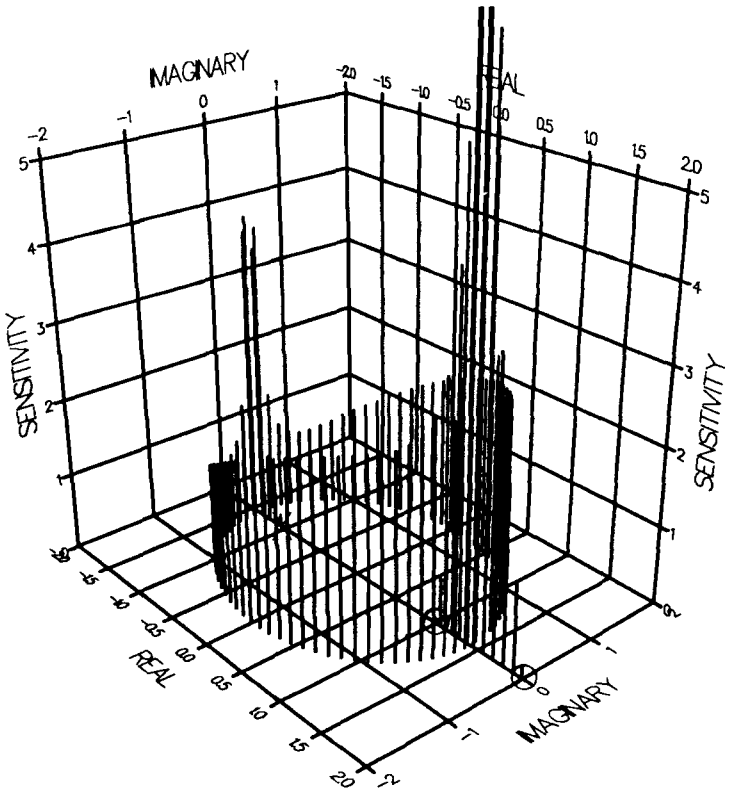
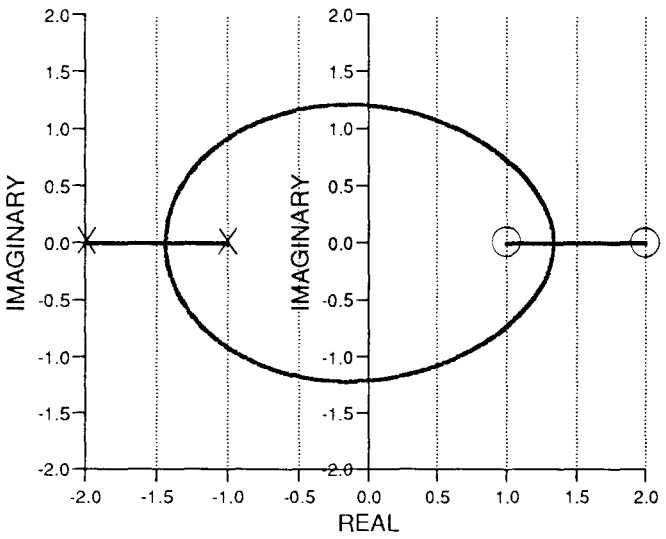
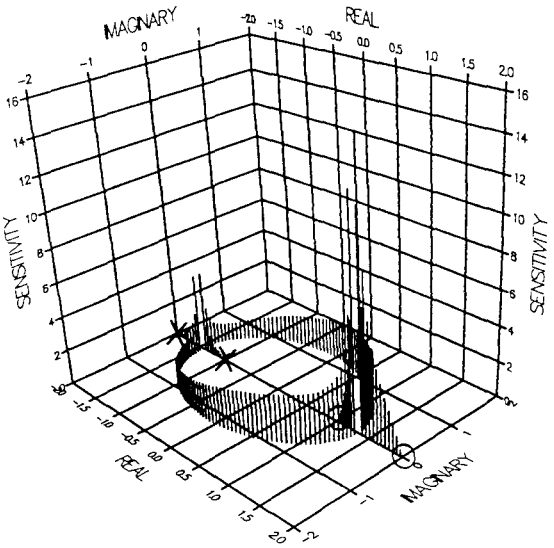


FIGURE 4.2 L SECOND-ORDER PLANT

ROOT LOCUS

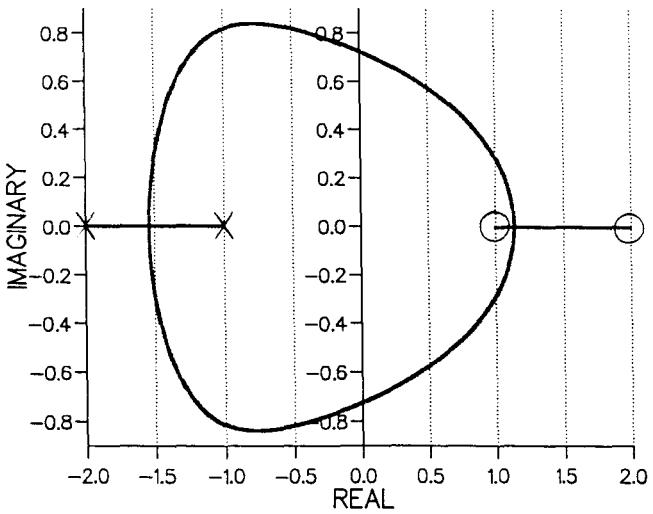


3-D SENSITIVITY PROFILE

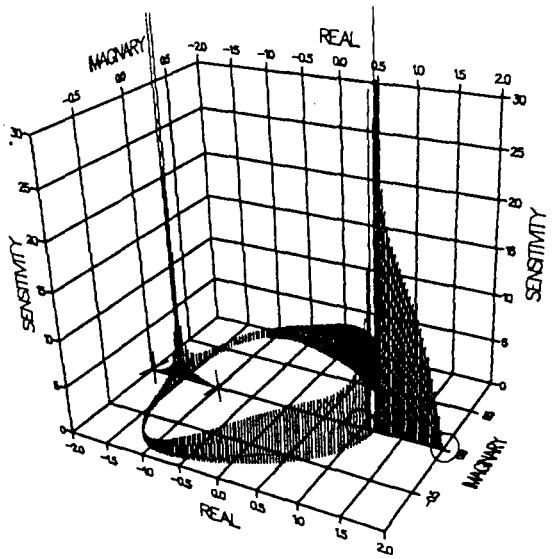


0.2 sec Delay

ROOT LOCUS



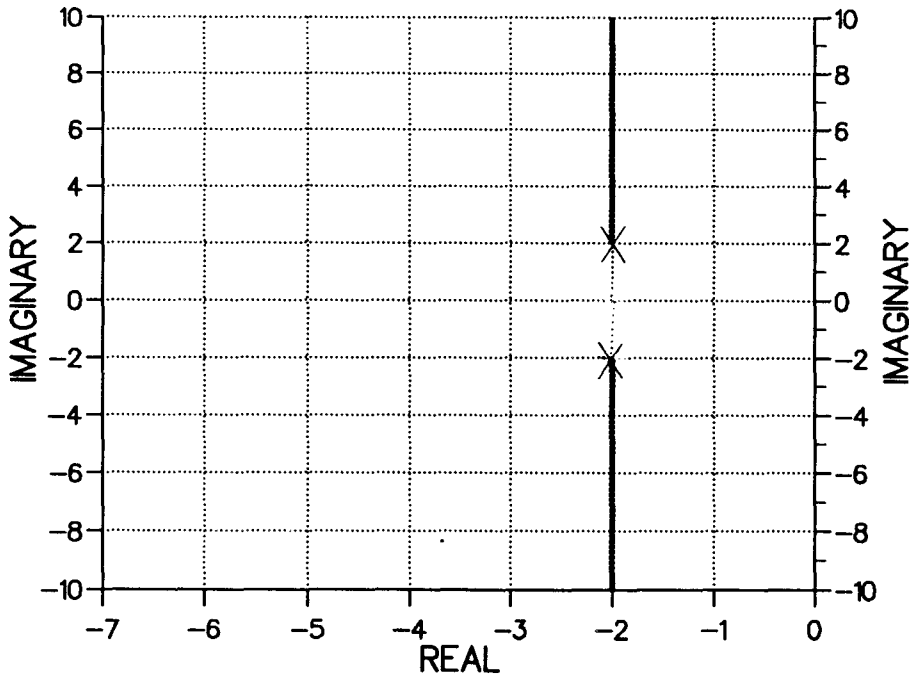
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 1 SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

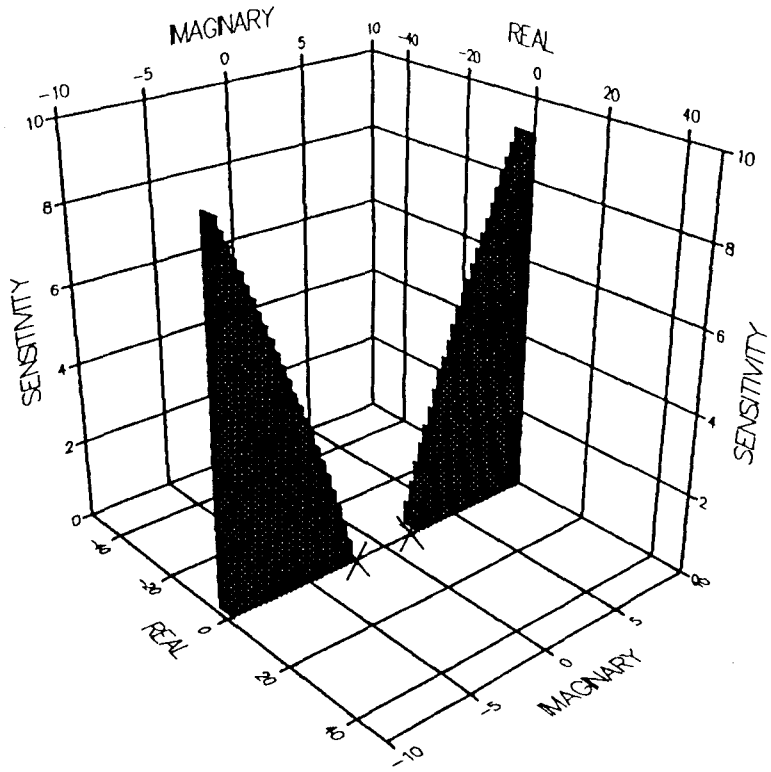
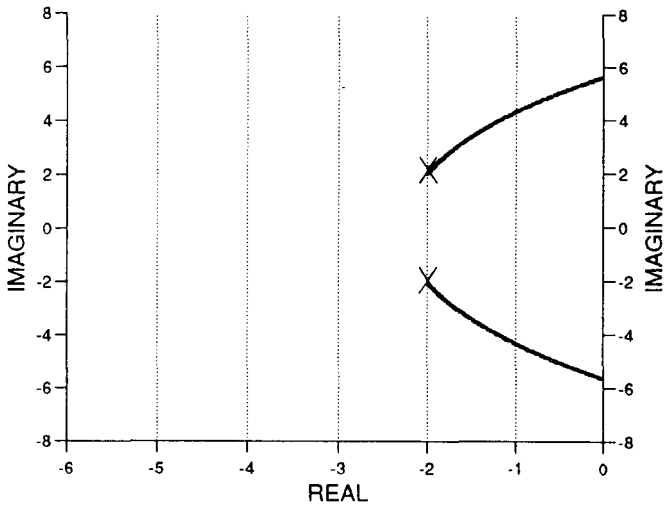
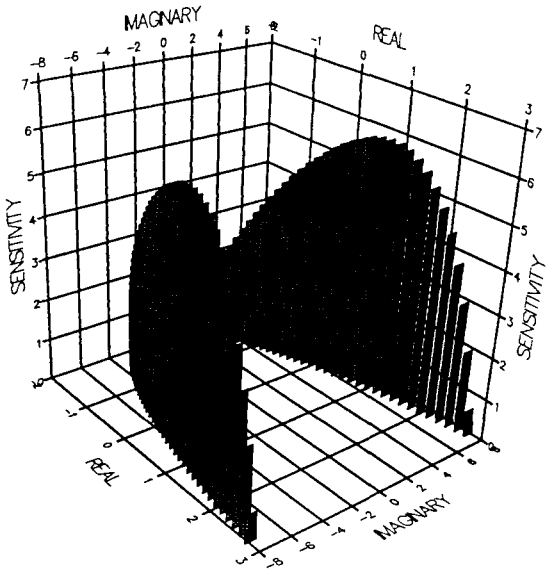


FIGURE 4.2 M SECOND-ORDER PLANT

ROOT LOCUS

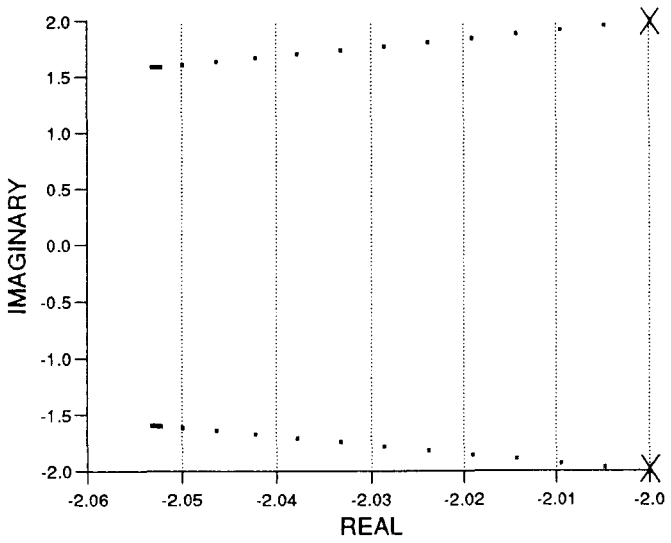


3-D SENSITIVITY PROFILE

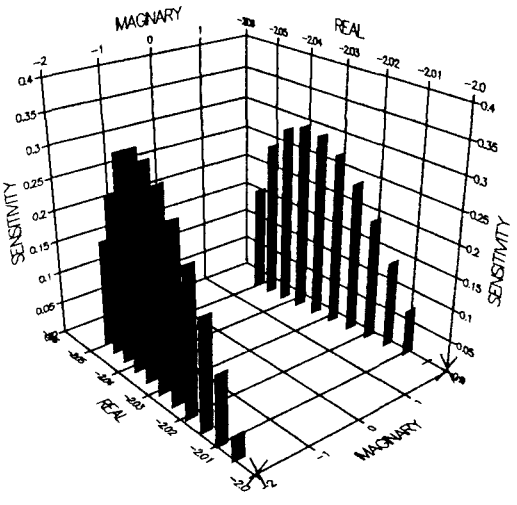


0.2 sec Delay

ROOT LOCUS



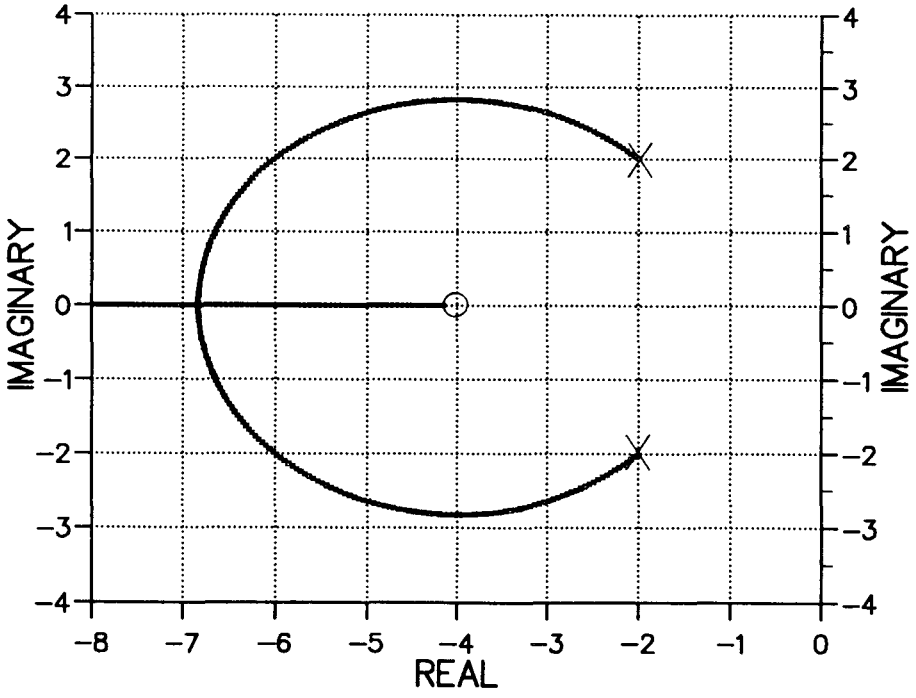
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 m SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

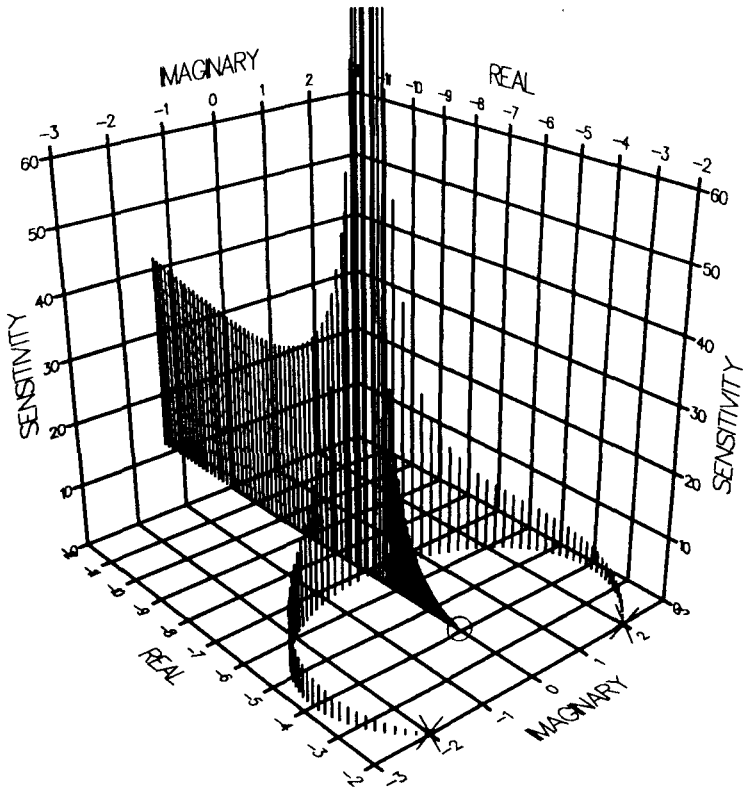
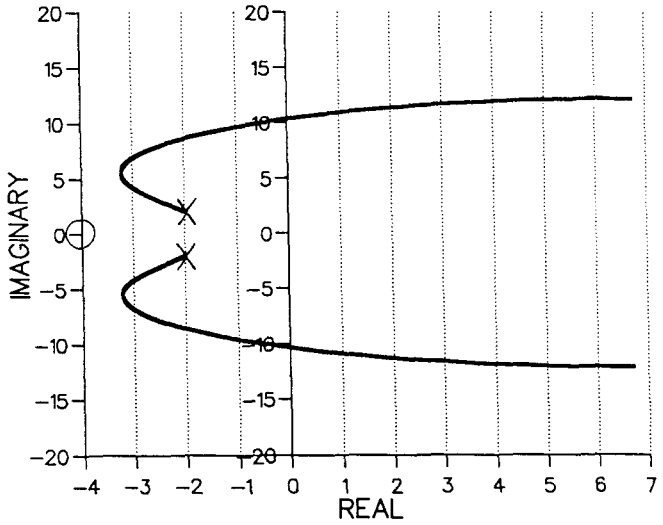
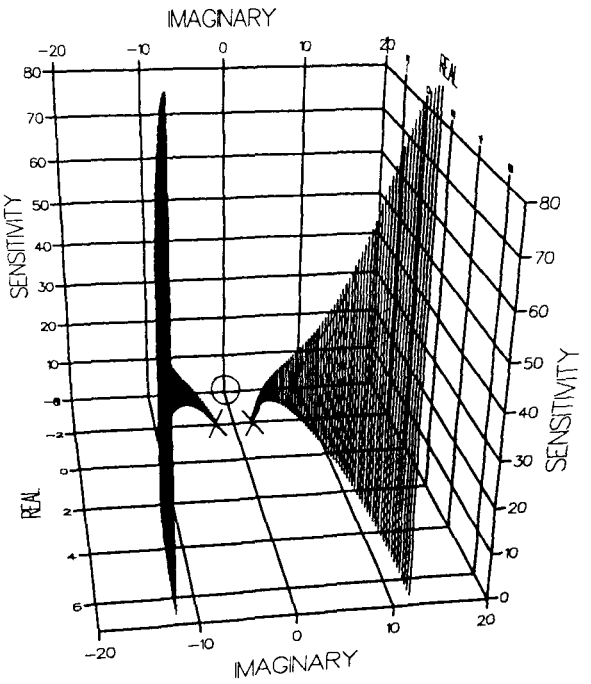


FIGURE 4.2 N SECOND-ORDER PLANT

ROOT LOCUS

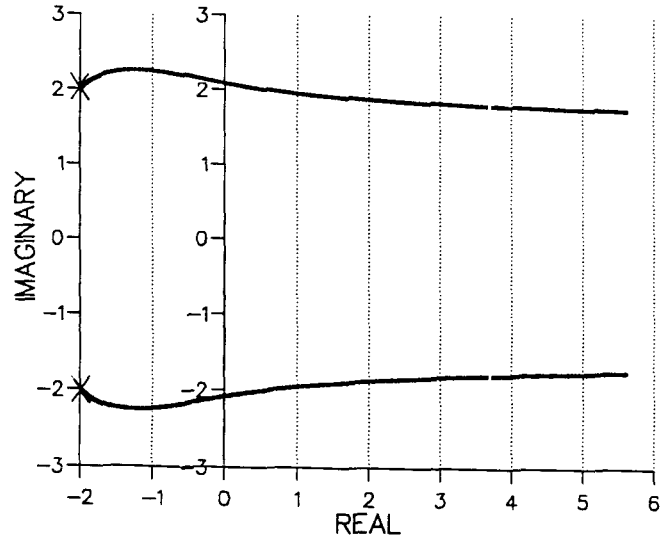


3-D SENSITIVITY PROFILE

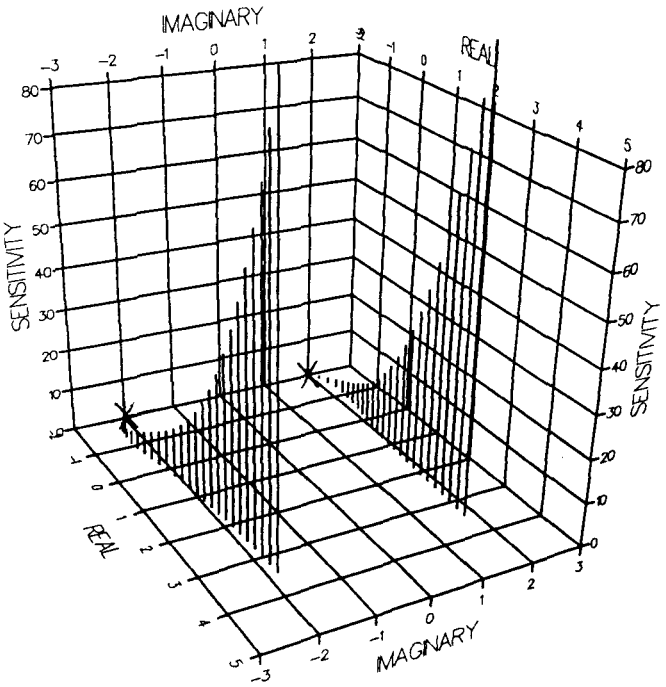


0.2 sec Delay

ROOT LOCUS



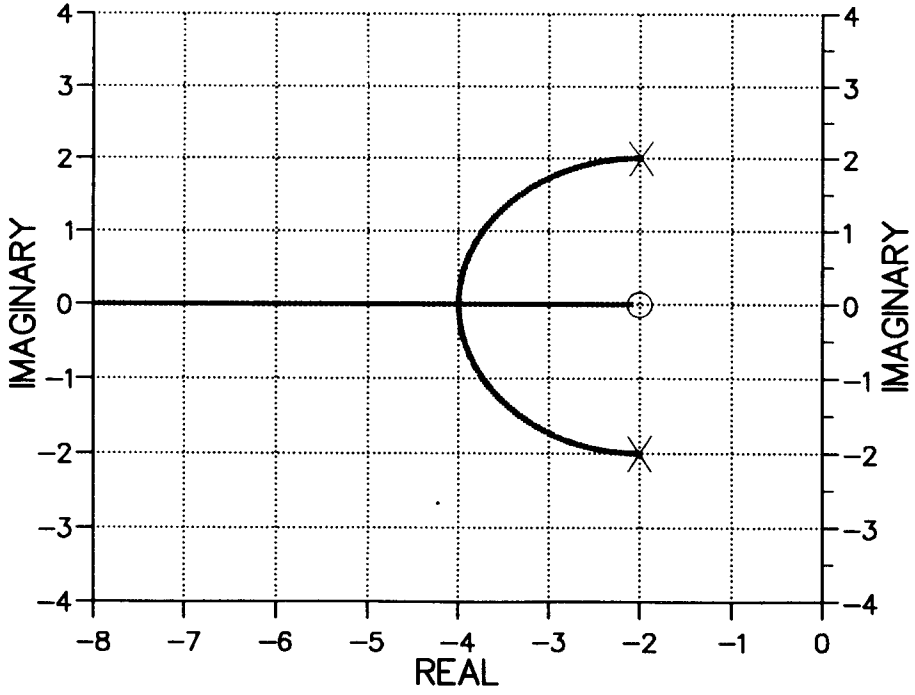
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 n SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

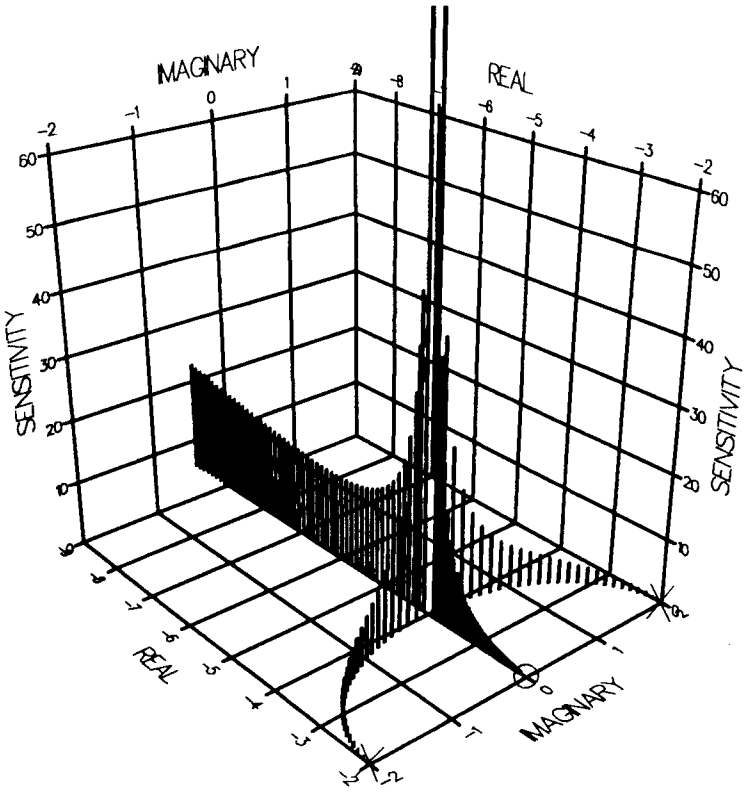
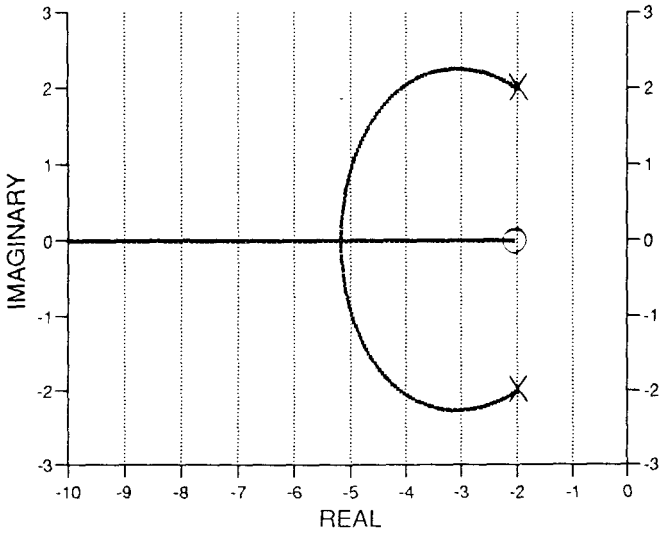
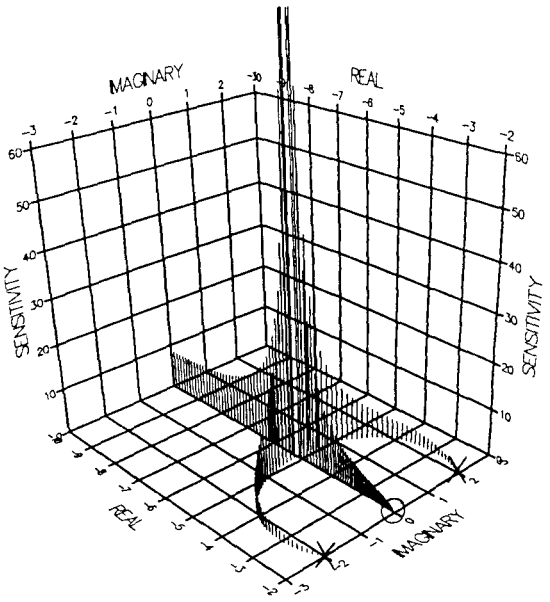


FIGURE 4.2 O SECOND-ORDER PLANT

ROOT LOCUS

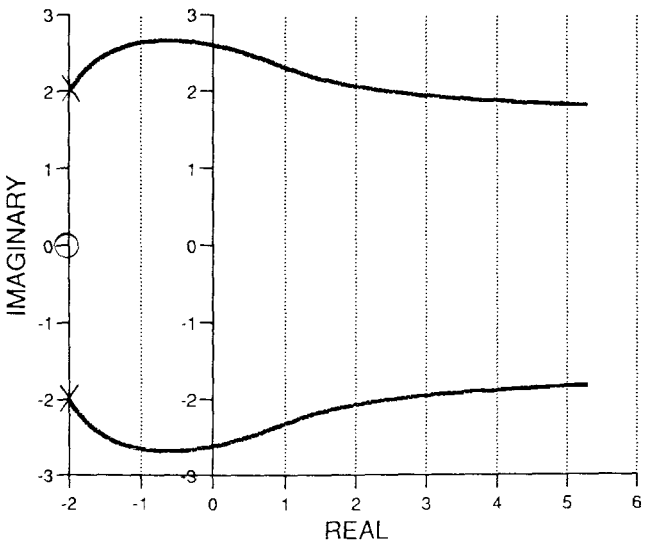


3-D SENSITIVITY PROFILE

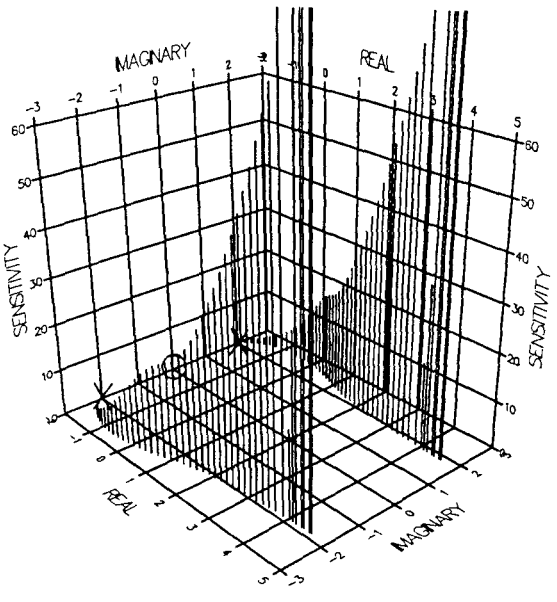


0.2 sec Delay

ROOT LOCUS



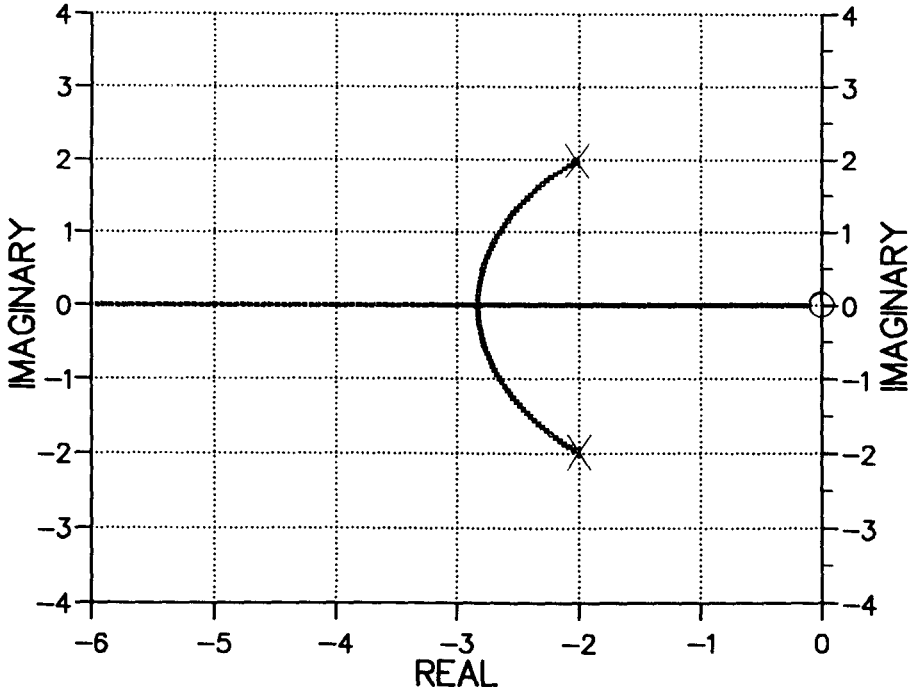
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 o SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

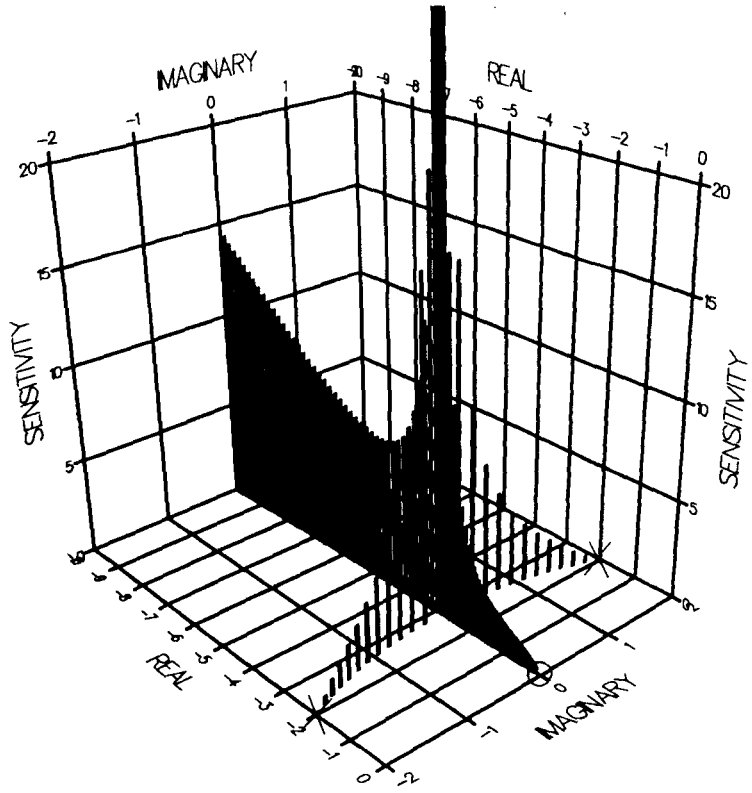
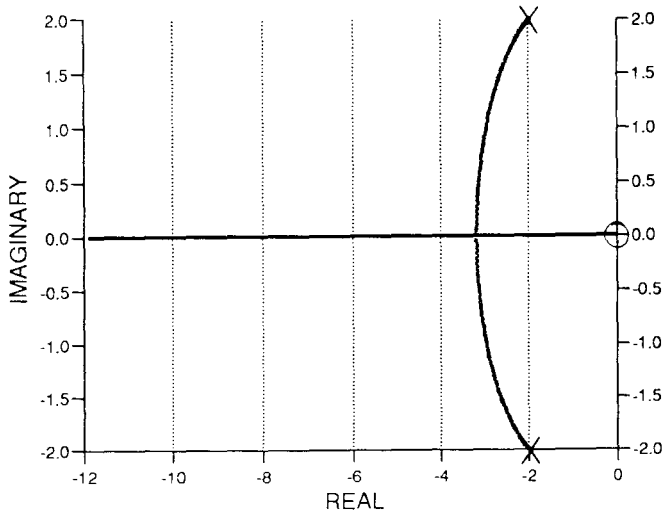
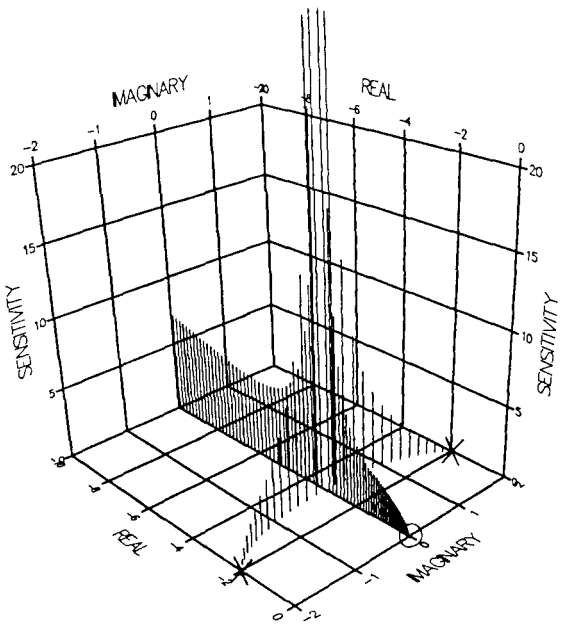


FIGURE 4.2 P SECOND-ORDER PLANT

ROOT LOCUS

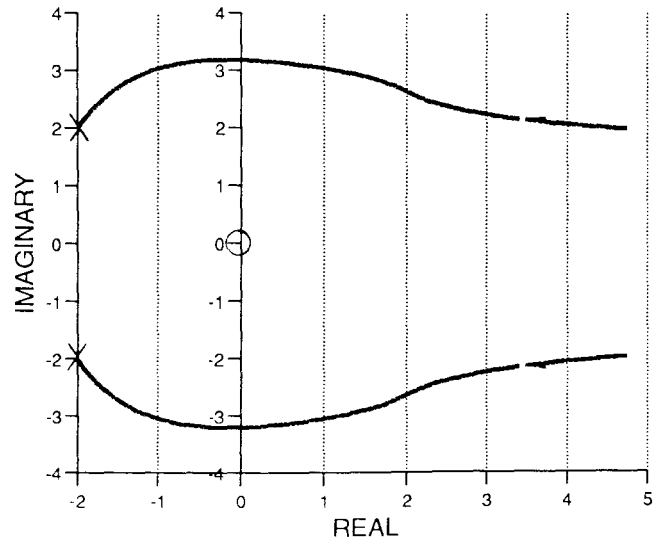


3-D SENSITIVITY PROFILE

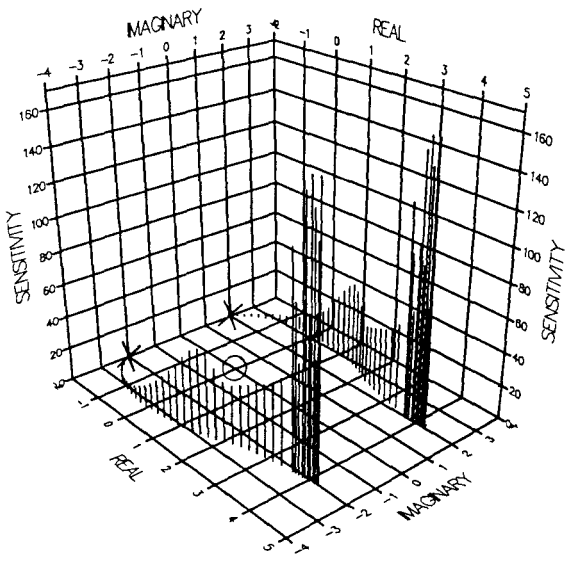


0.2 sec Delay

ROOT LOCUS



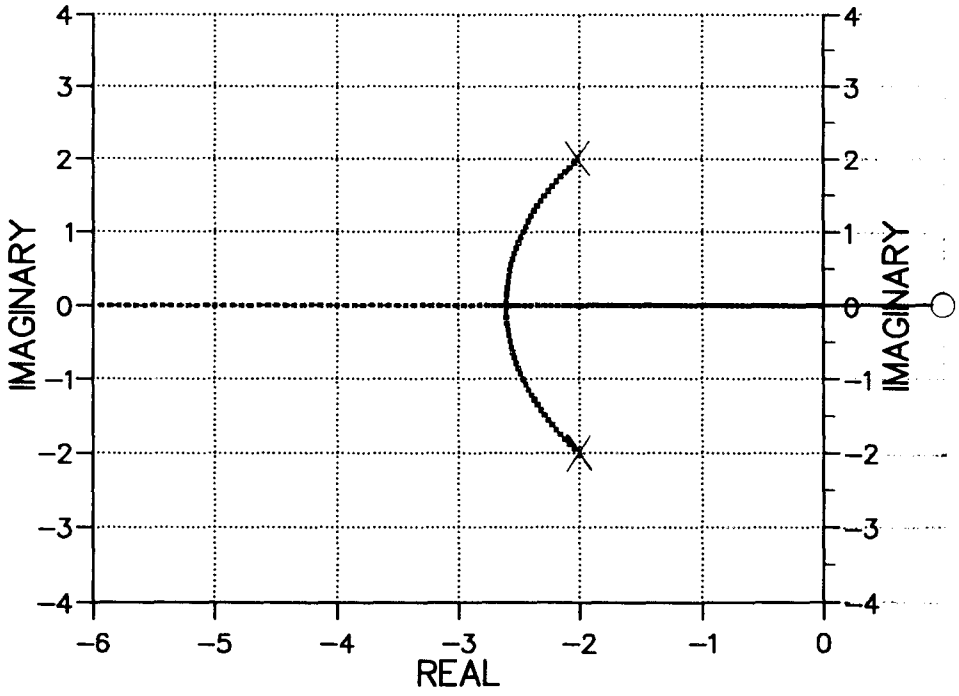
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 p SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

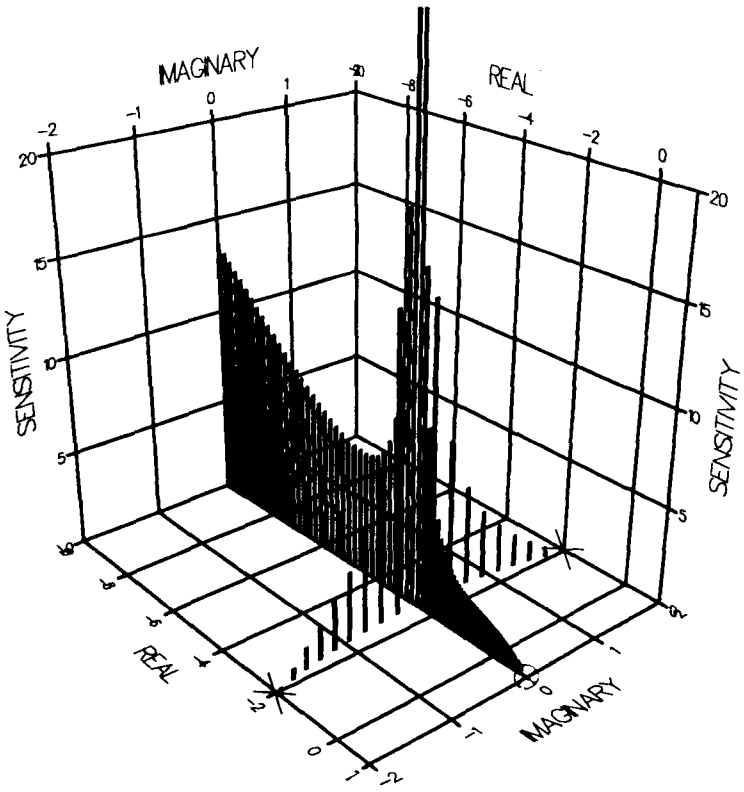
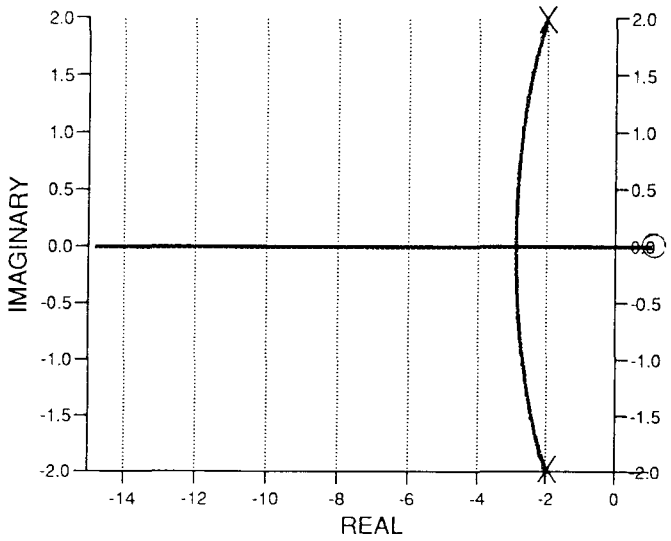
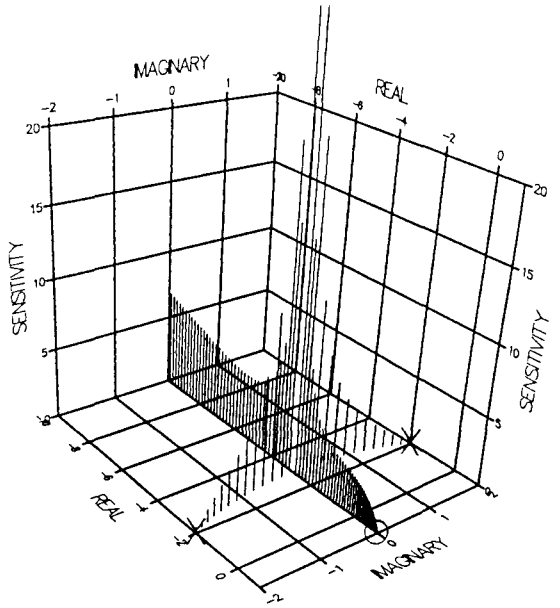


FIGURE 4.2 Q SECOND-ORDER PLANT

ROOT LOCUS

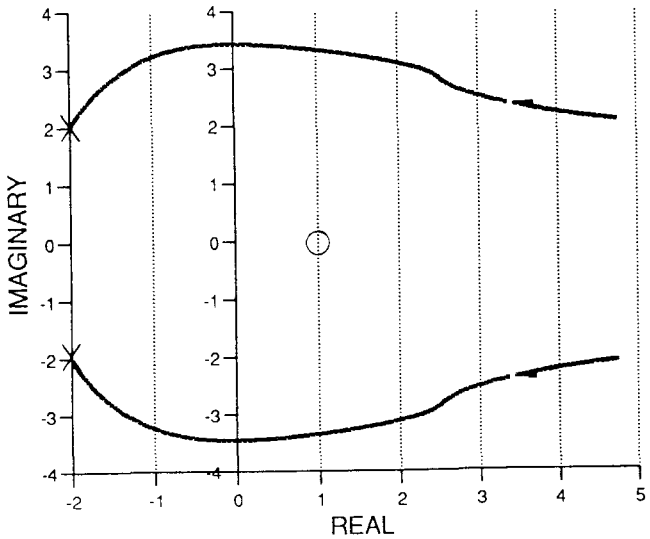


3-D SENSITIVITY PROFILE

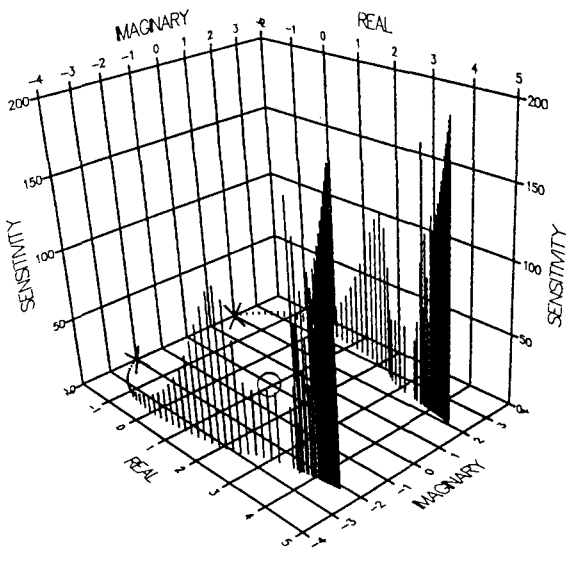


0.2 sec Delay

ROOT LOCUS



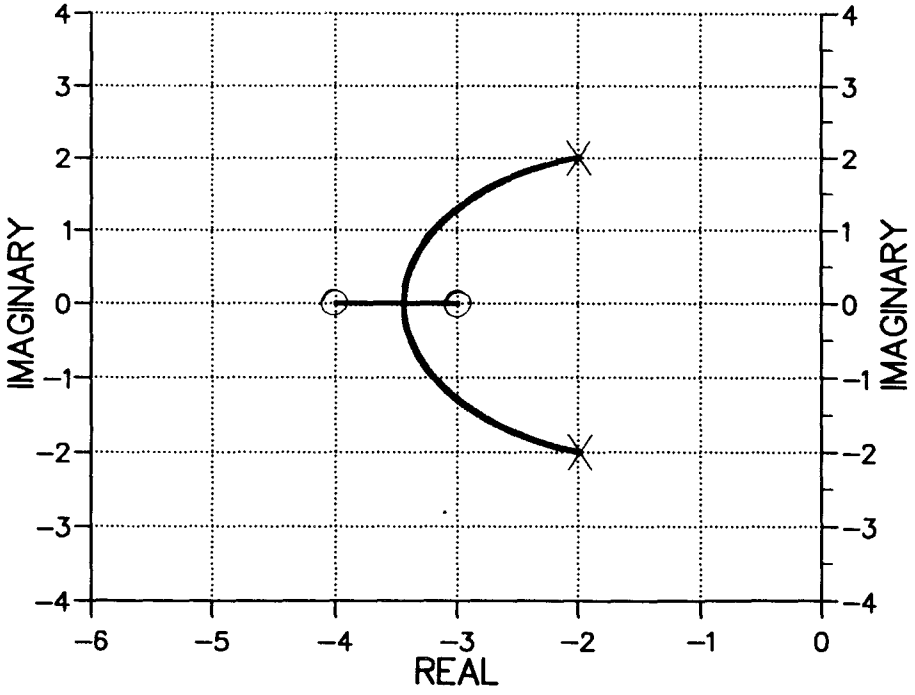
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 q SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

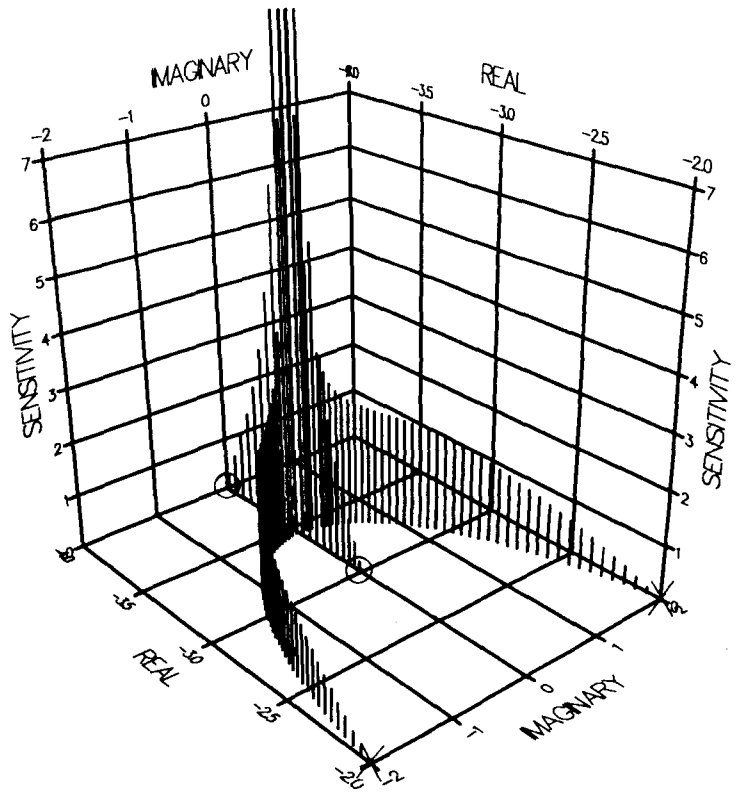
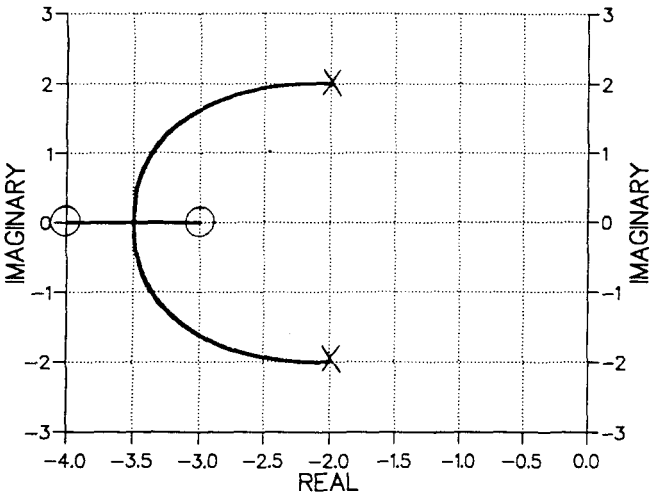
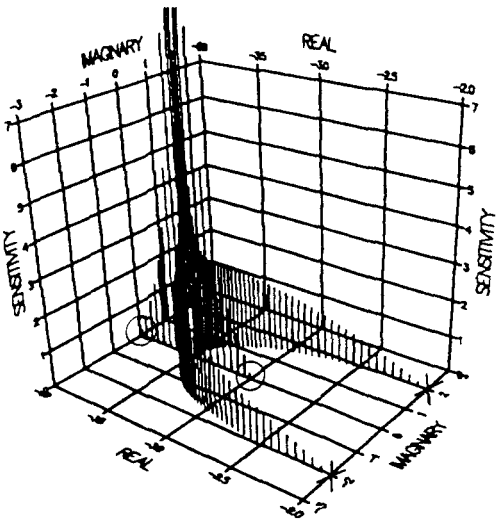


FIGURE 4.2 R SECOND-ORDER PLANT

ROOT LOCUS

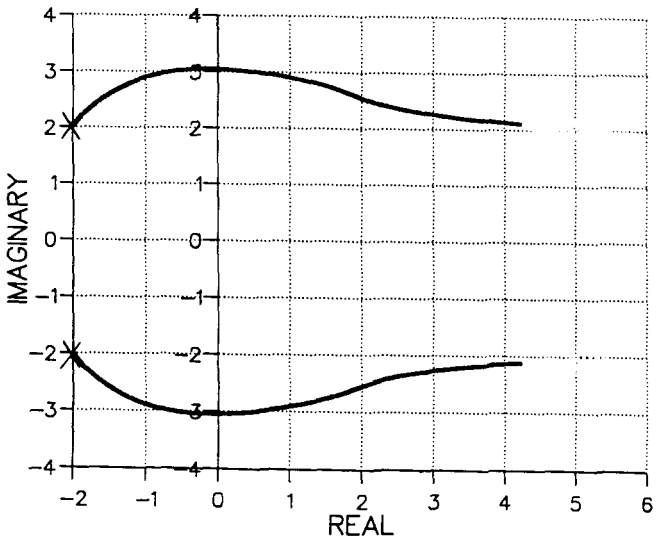


3-D SENSITIVITY PROFILE

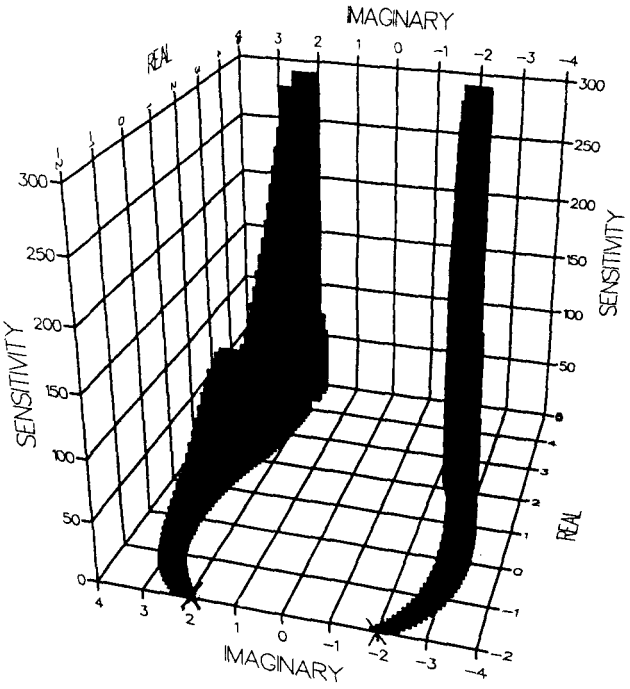


0.2 sec Delay

ROOT LOCUS



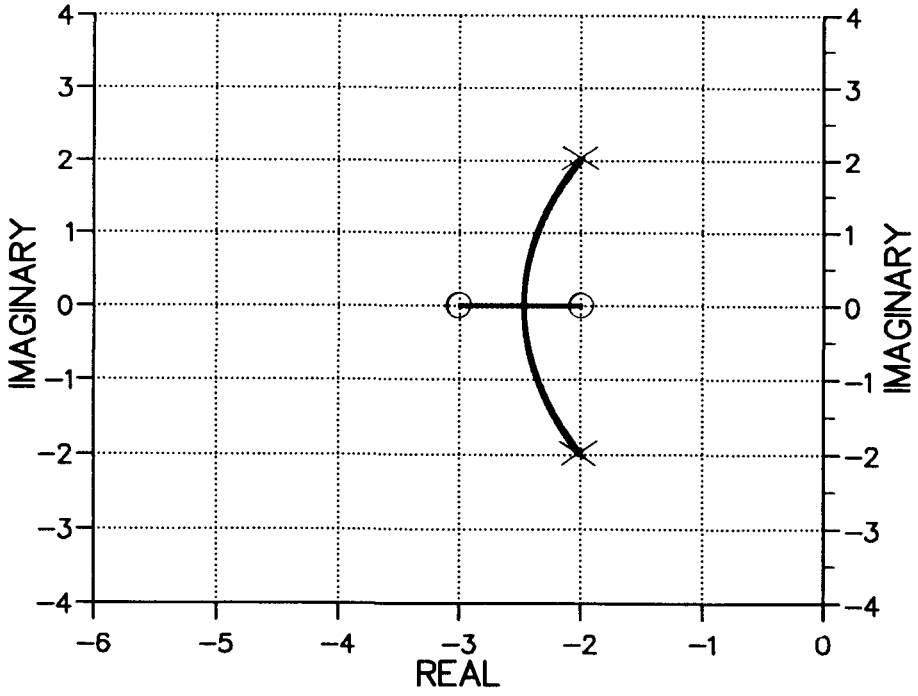
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 r SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

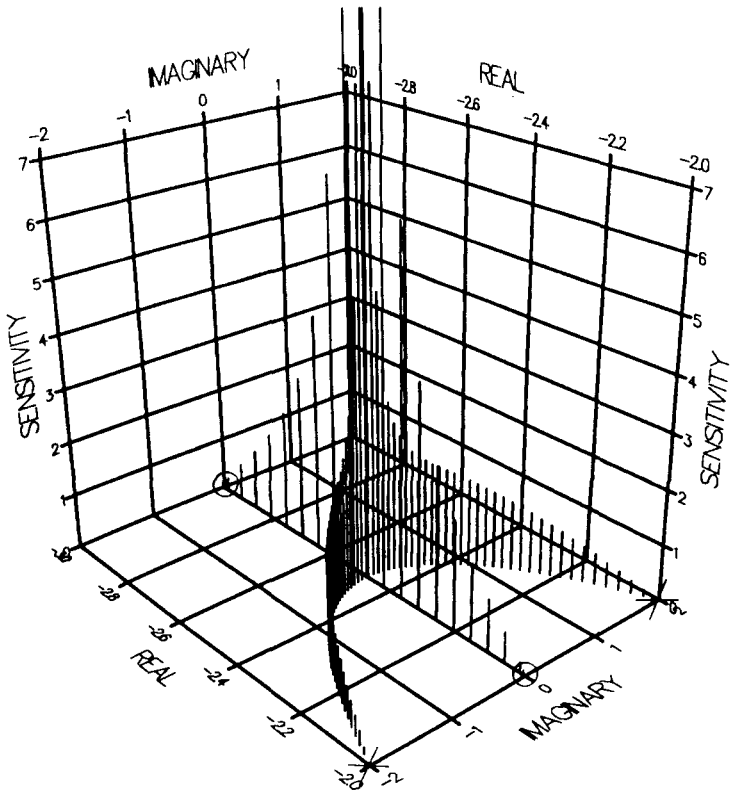
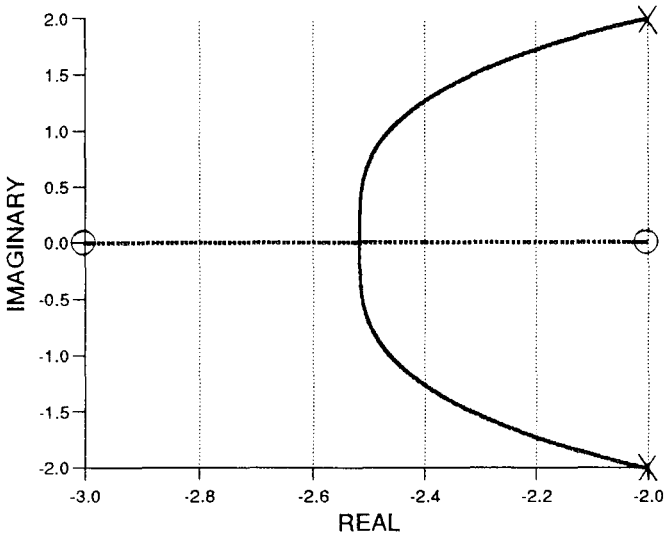
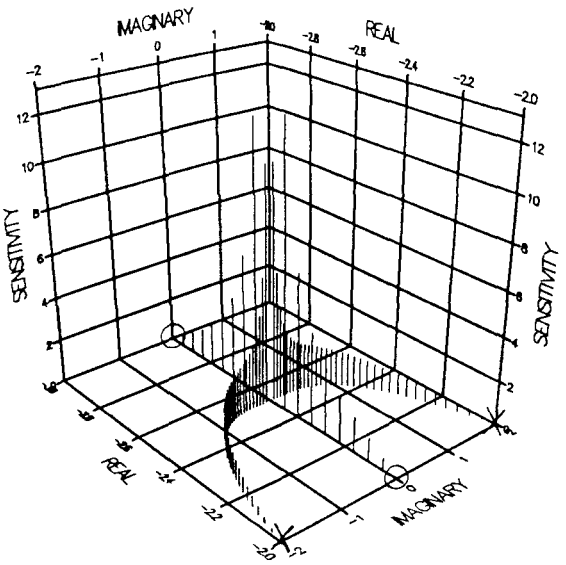


FIGURE 4.2 S SECOND-ORDER PLANT

ROOT LOCUS

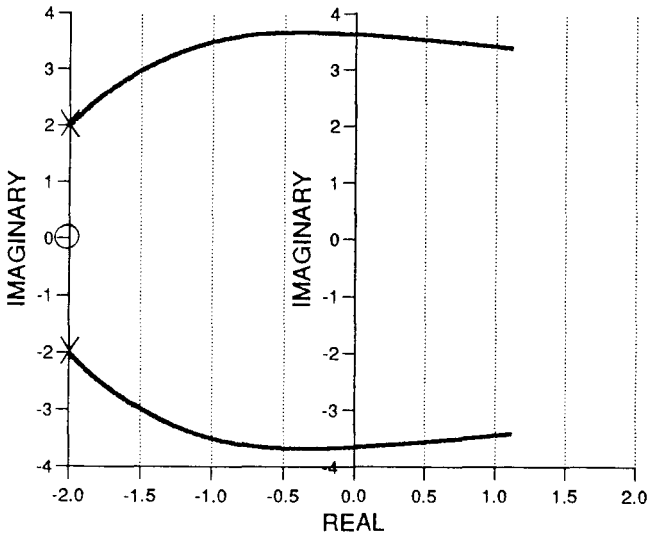


3-D SENSITIVITY PROFILE

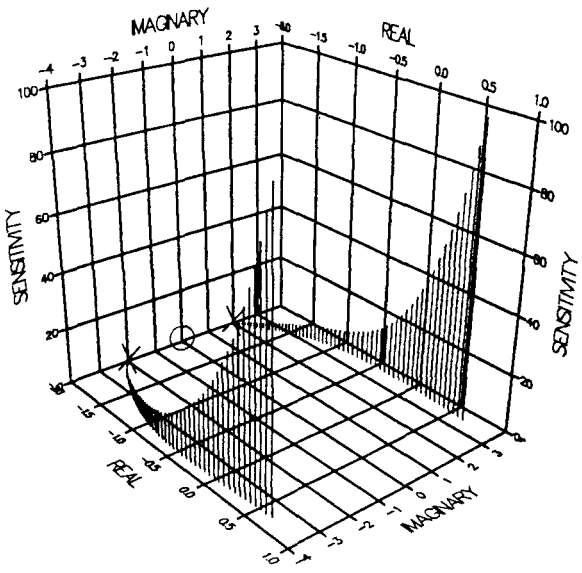


0.2 sec Delay

ROOT LOCUS



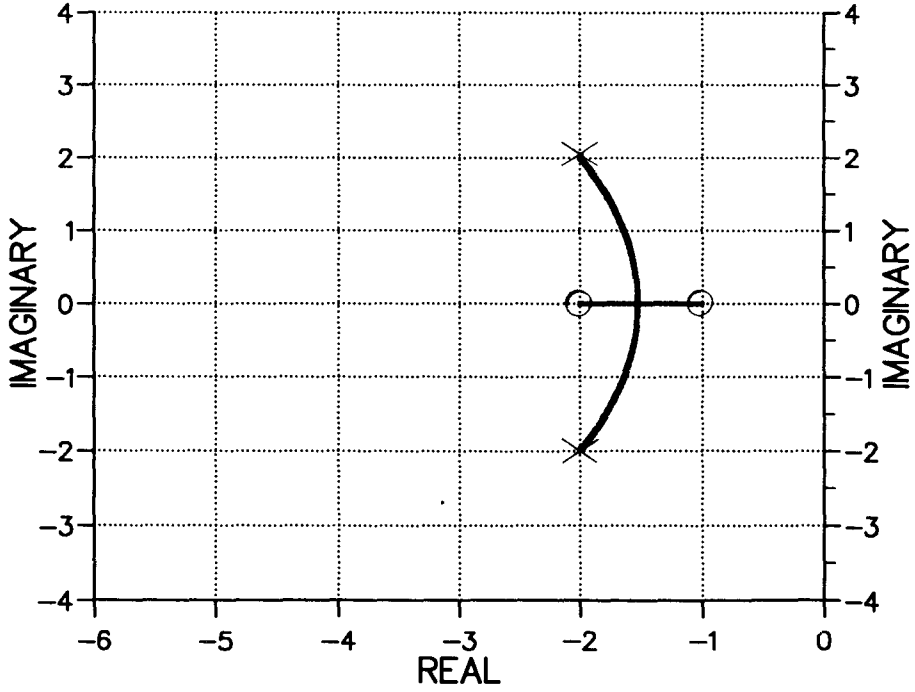
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 s SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

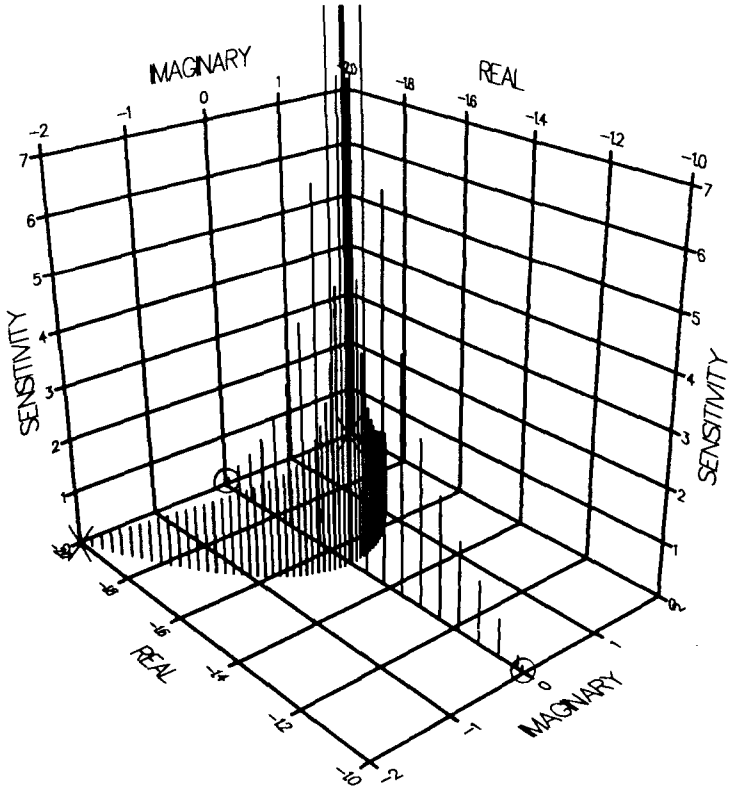
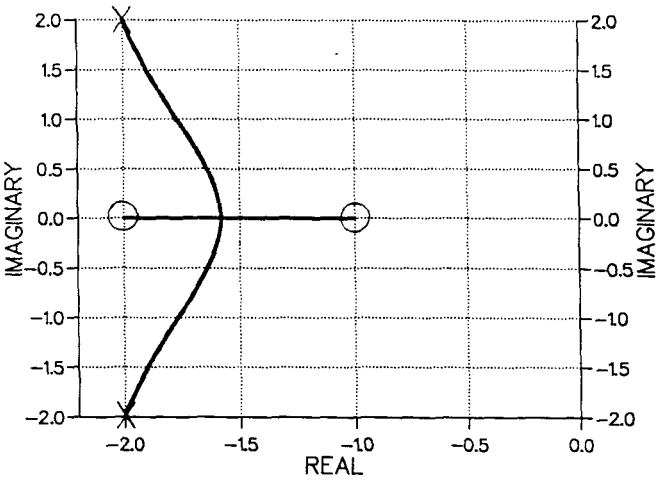
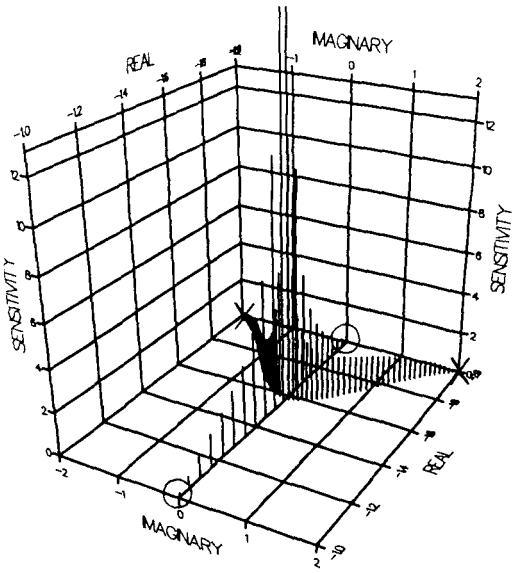


FIGURE 4.2 T SECOND-ORDER PLANT

ROOT LOCUS

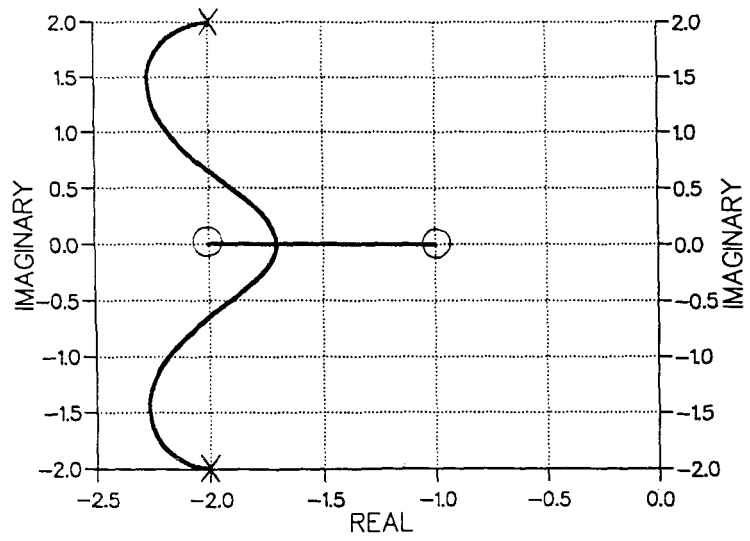


3-D SENSITIVITY PROFILE

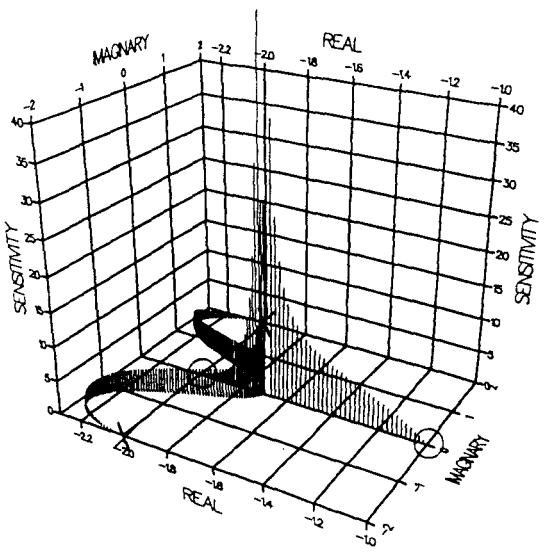


0.2 sec Delay

ROOT LOCUS



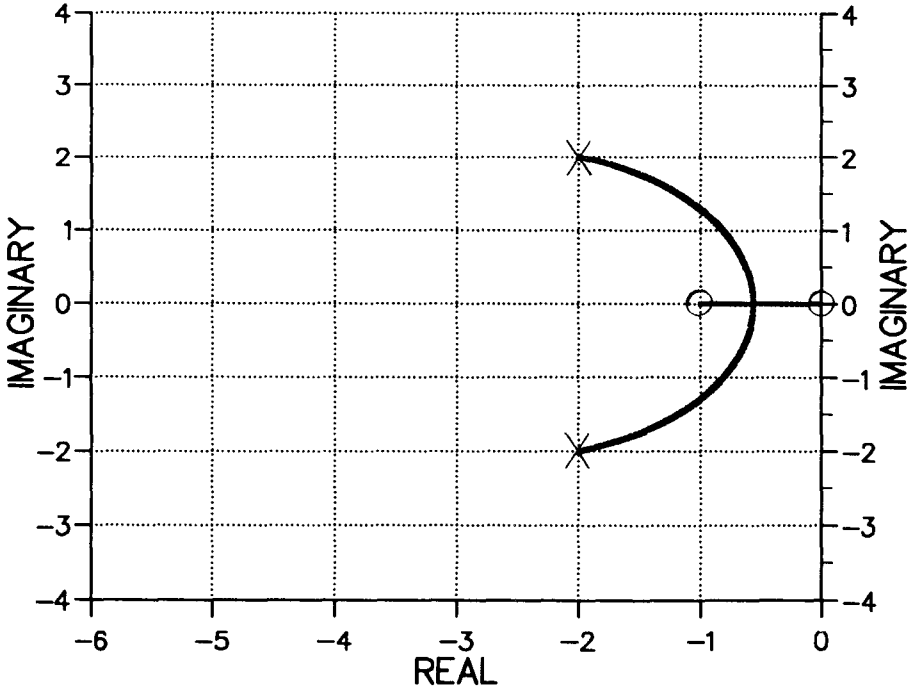
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 t SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

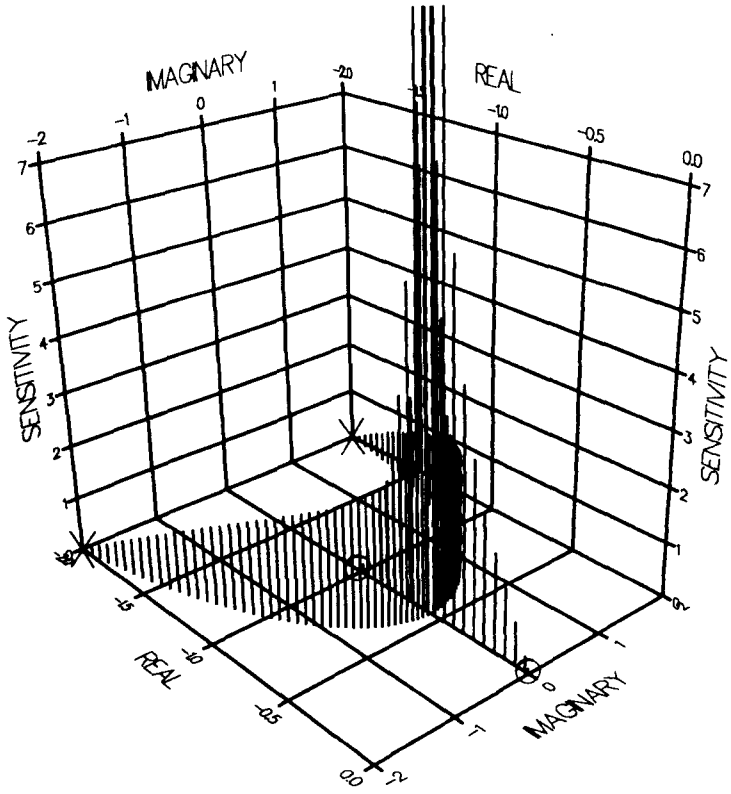
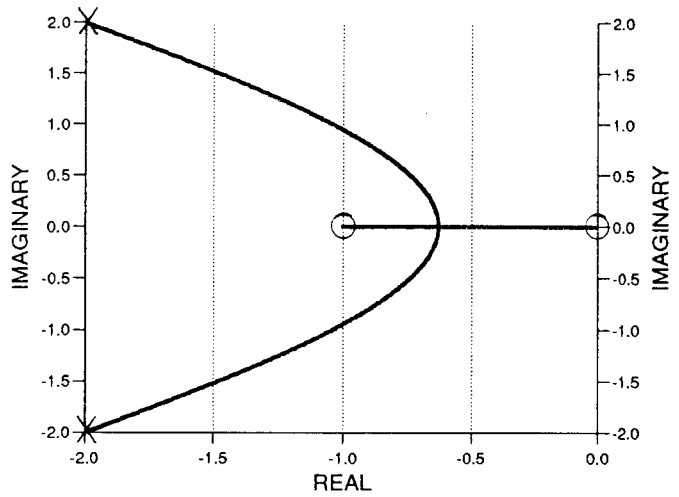
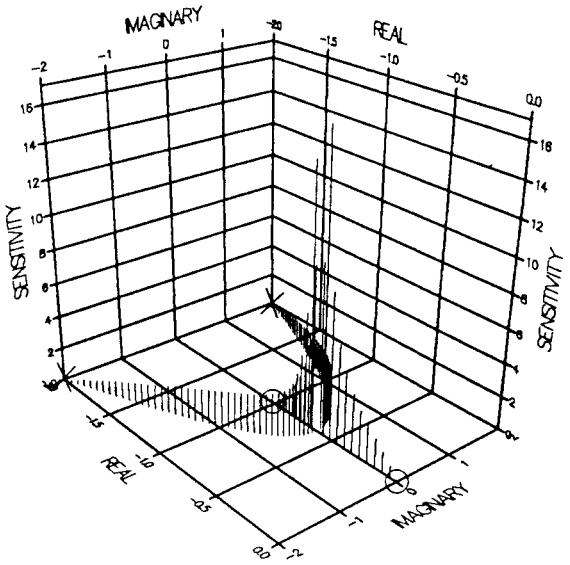


FIGURE 4.2 U SECOND-ORDER PLANT

ROOT LOCUS

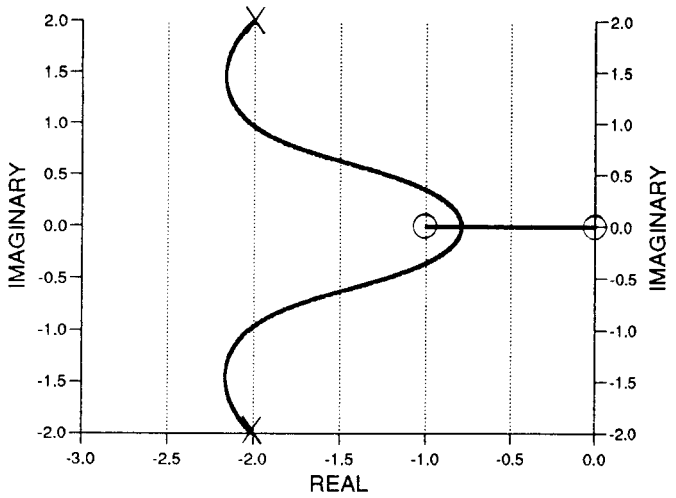


3-D SENSITIVITY PROFILE

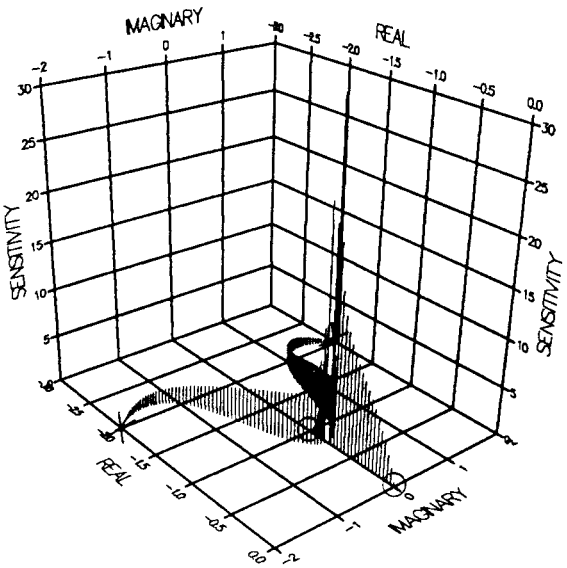


0.2 sec Delay

ROOT LOCUS



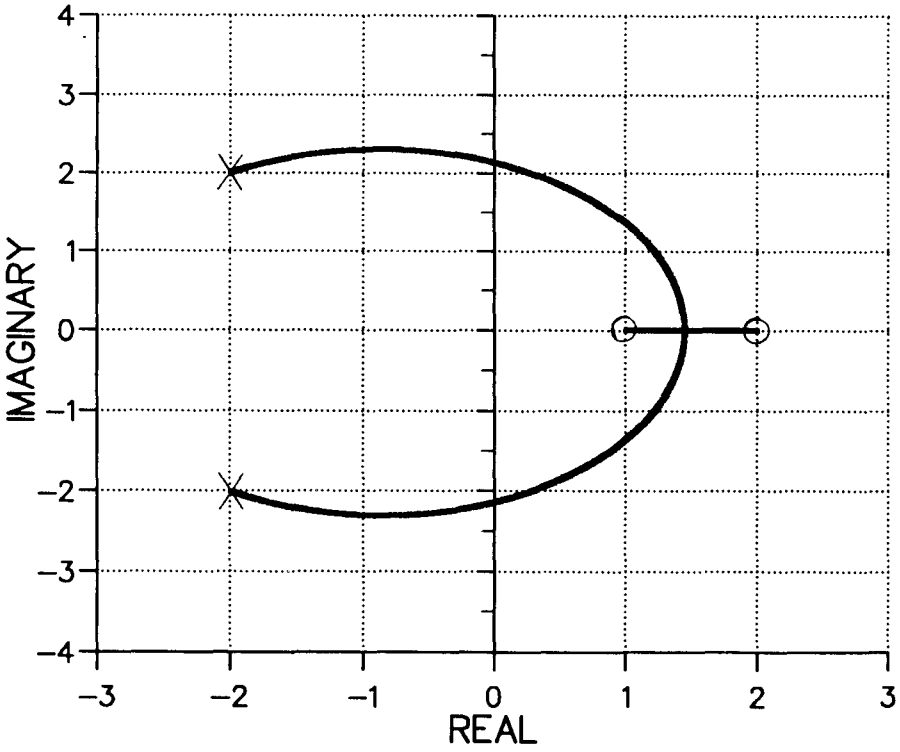
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 u SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

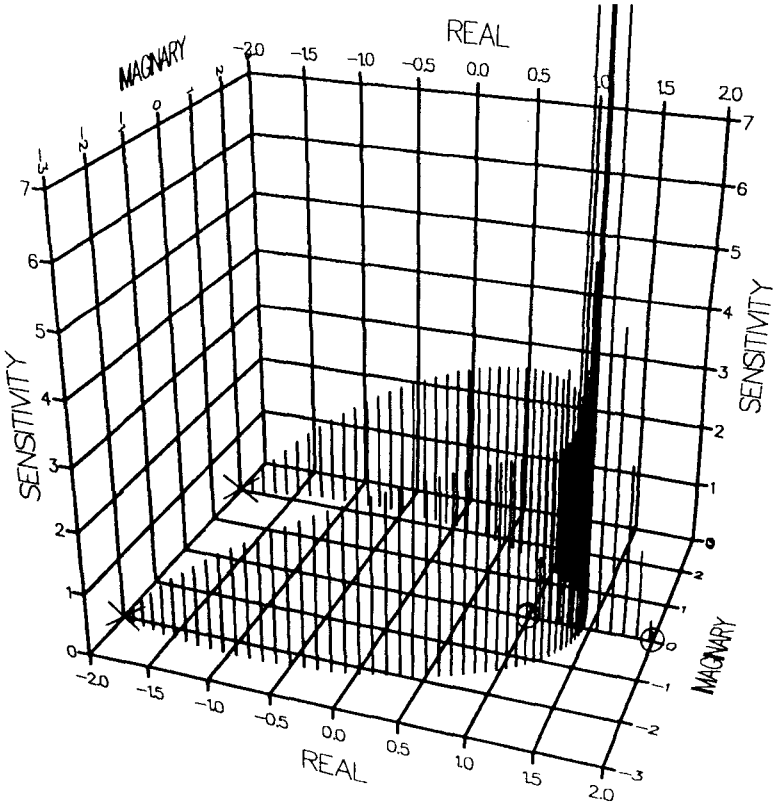
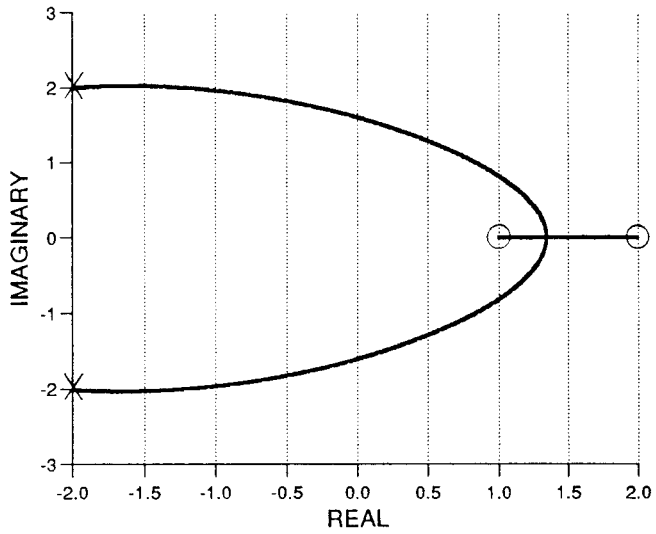
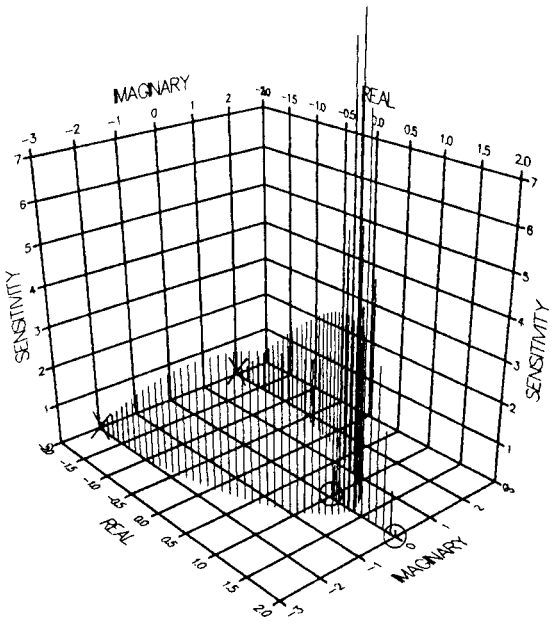


FIGURE 4.2 V SECOND-ORDER PLANT

ROOT LOCUS

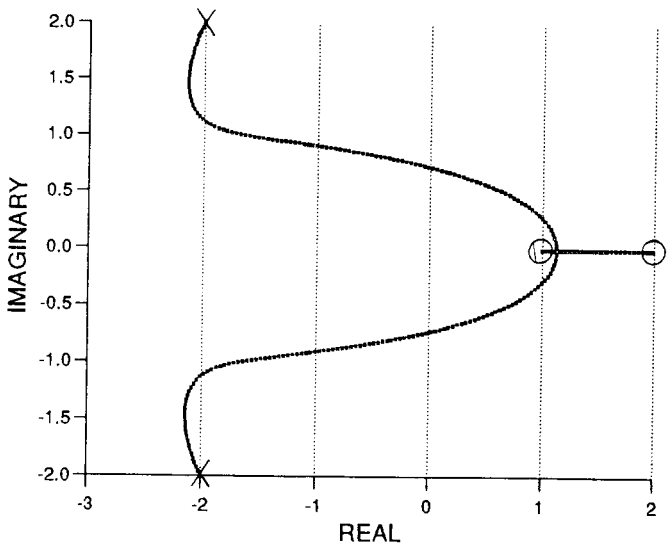


3-D SENSITIVITY PROFILE

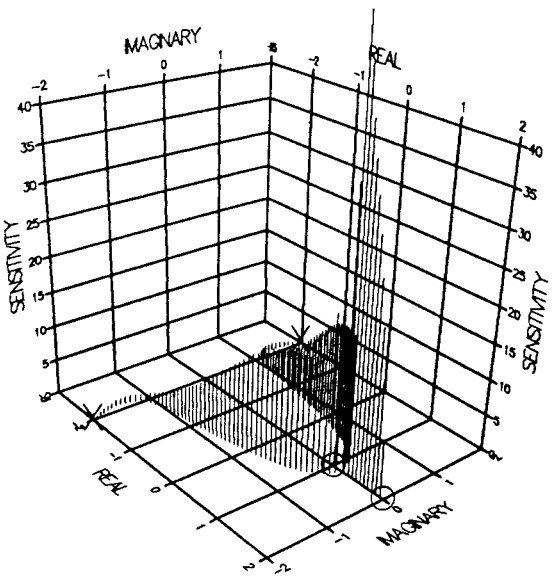


0.2 sec Delay

ROOT LOCUS



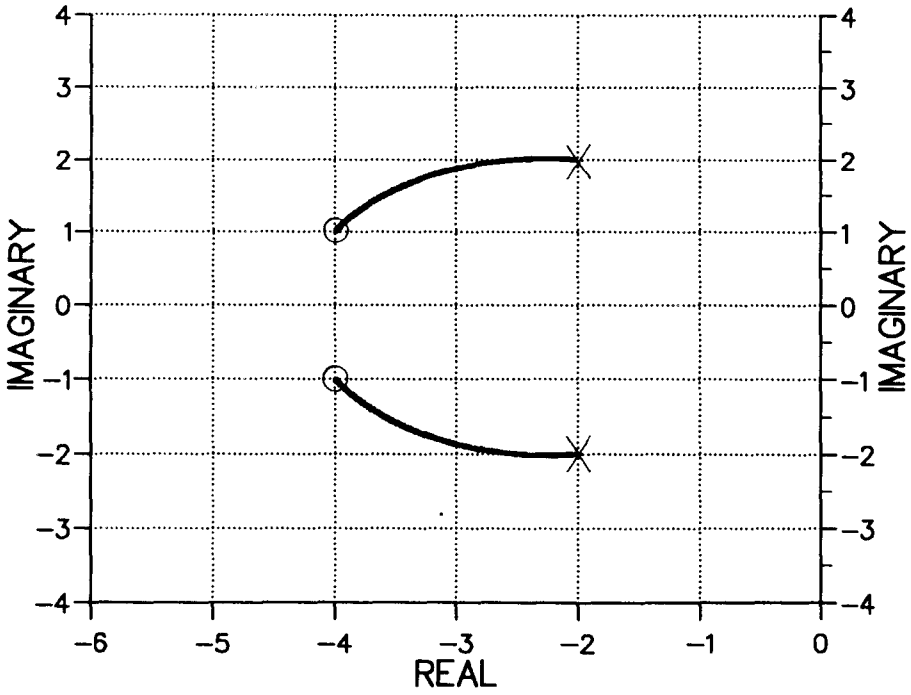
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 v SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

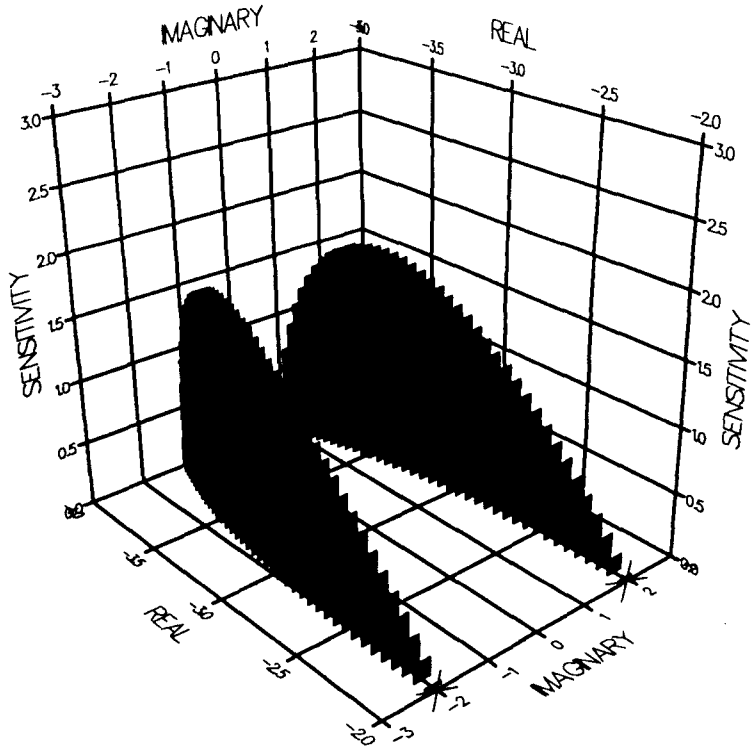
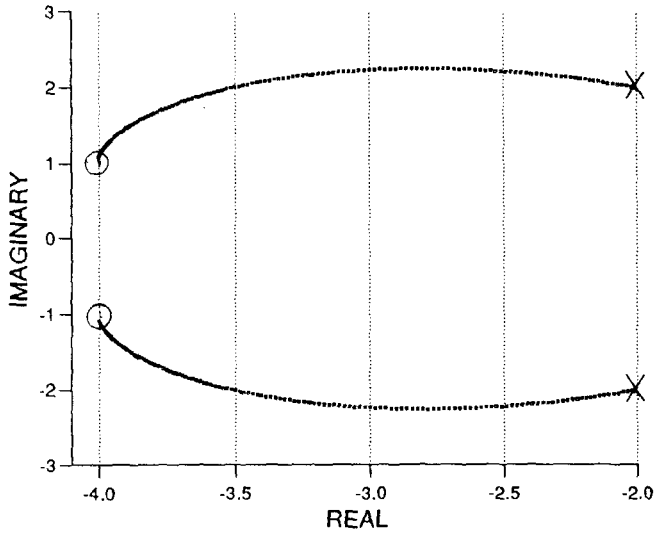
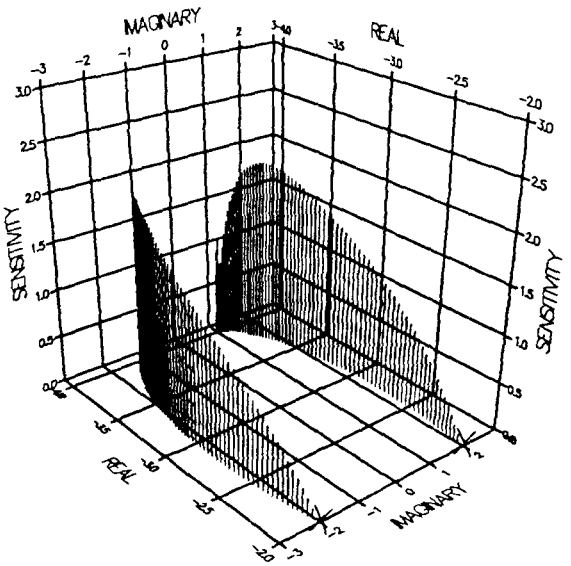


FIGURE 4.2 W SECOND-ORDER PLANT

ROOT LOCUS

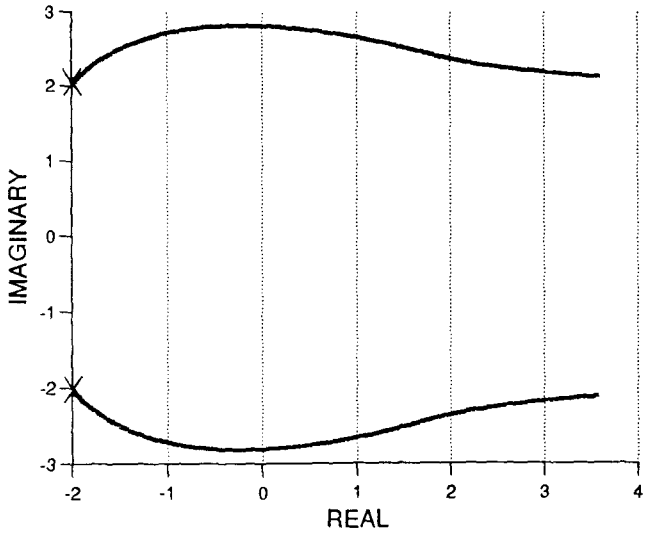


3-D SENSITIVITY PROFILE

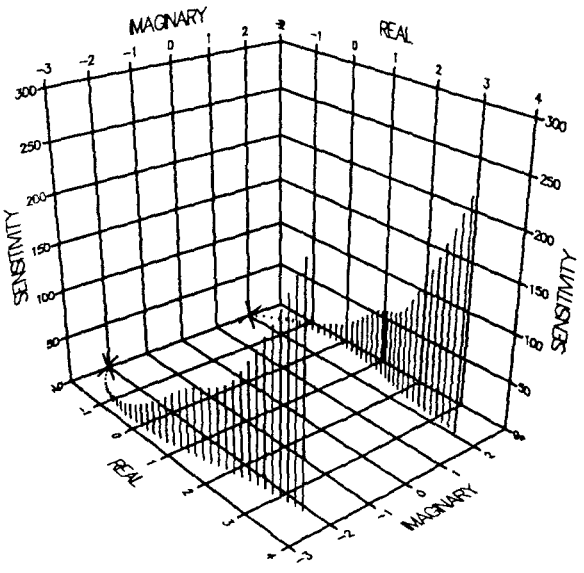


0.2 sec Delay

ROOT LOCUS



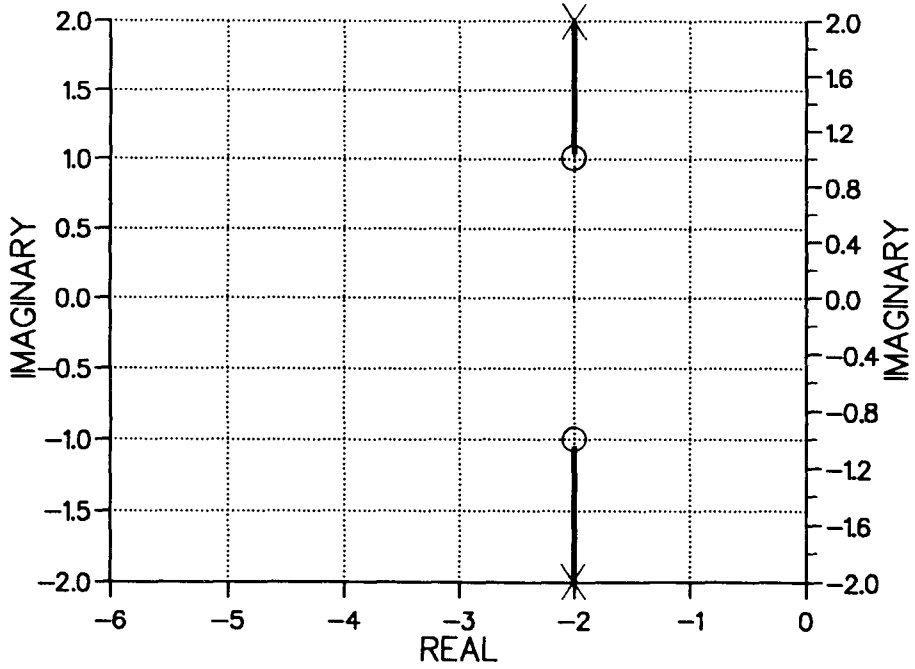
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 w SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

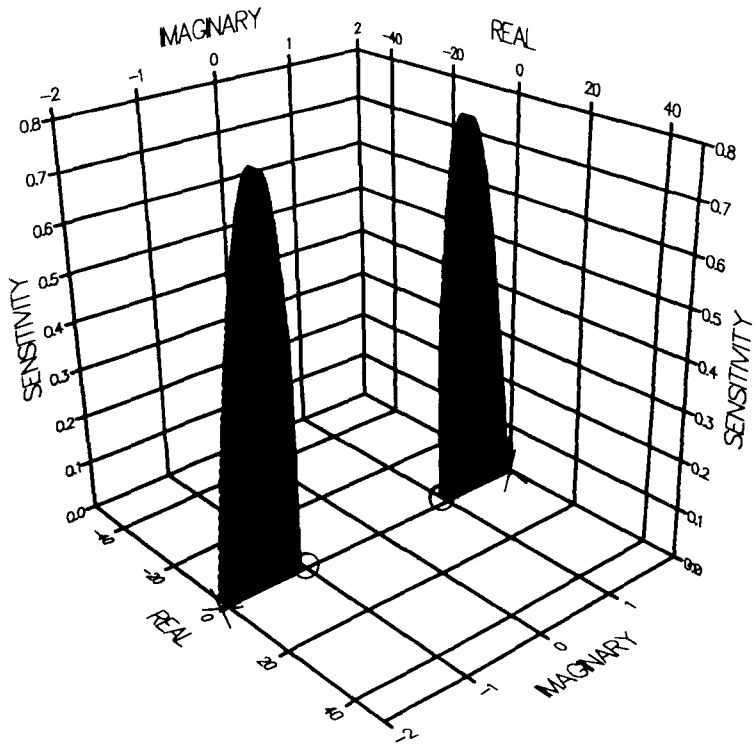
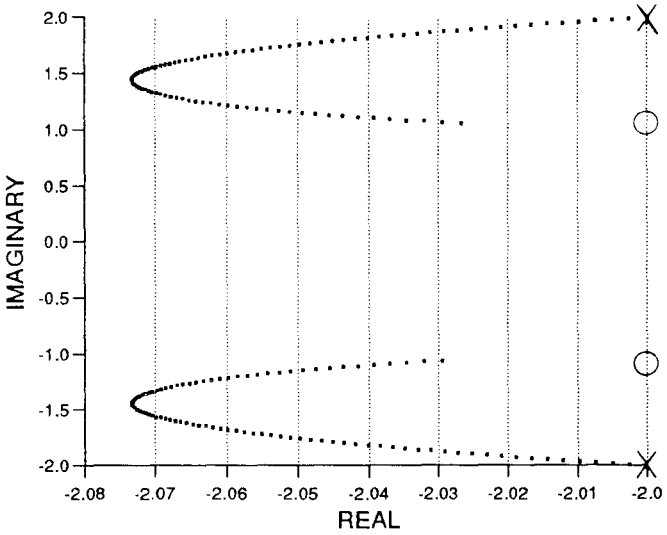
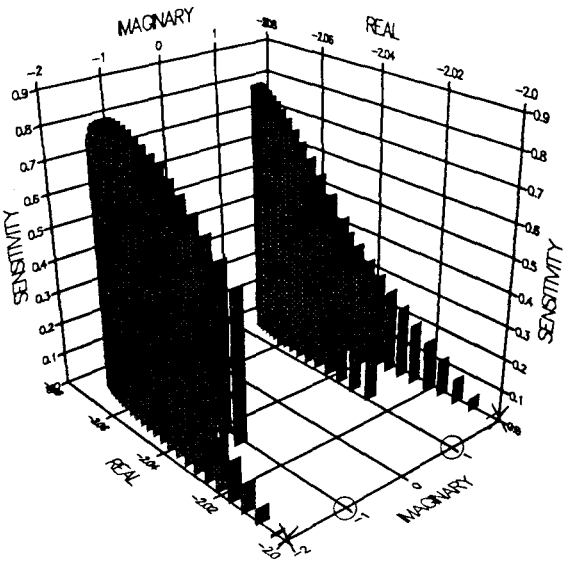


FIGURE 4.2 X SECOND-ORDER PLANT

ROOT LOCUS

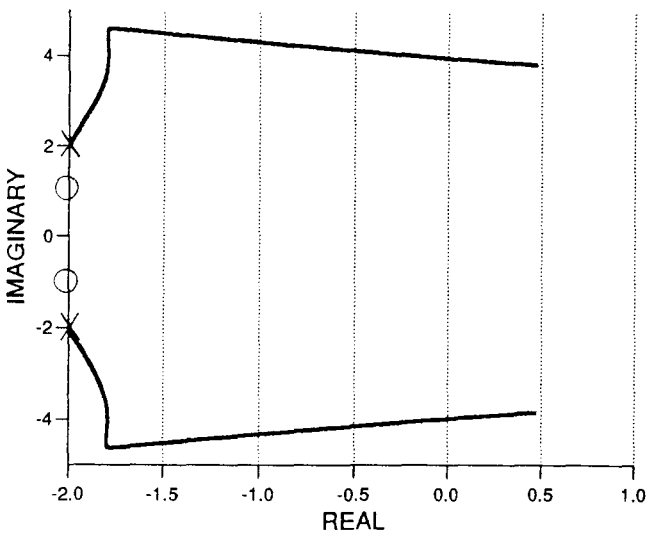


3-D SENSITIVITY PROFILE

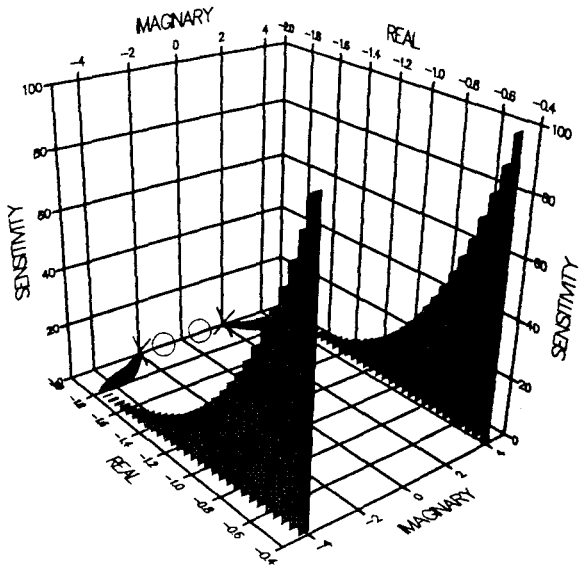


0.2 sec Delay

ROOT LOCUS



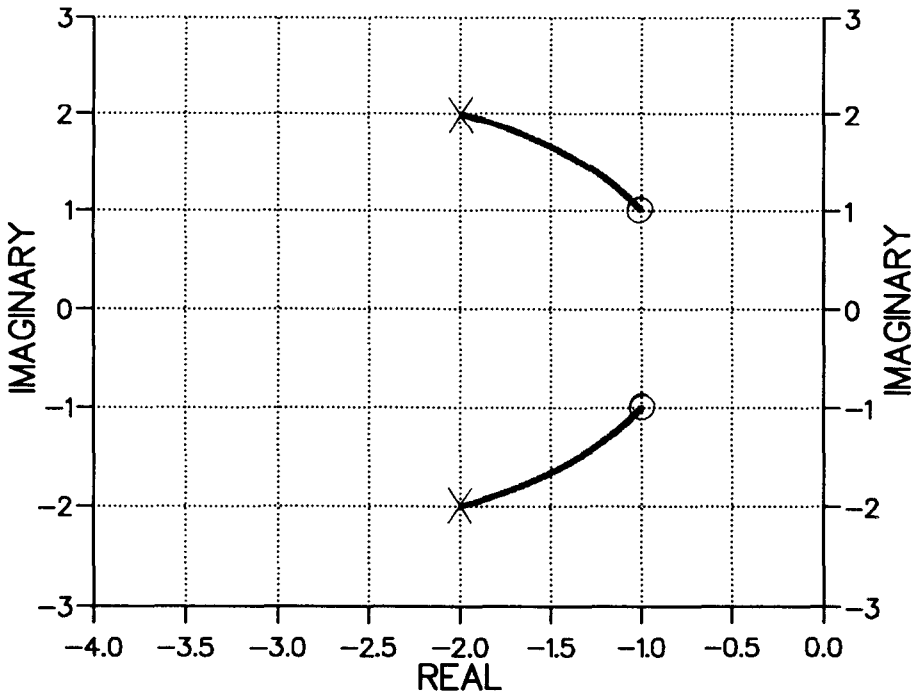
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 x SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

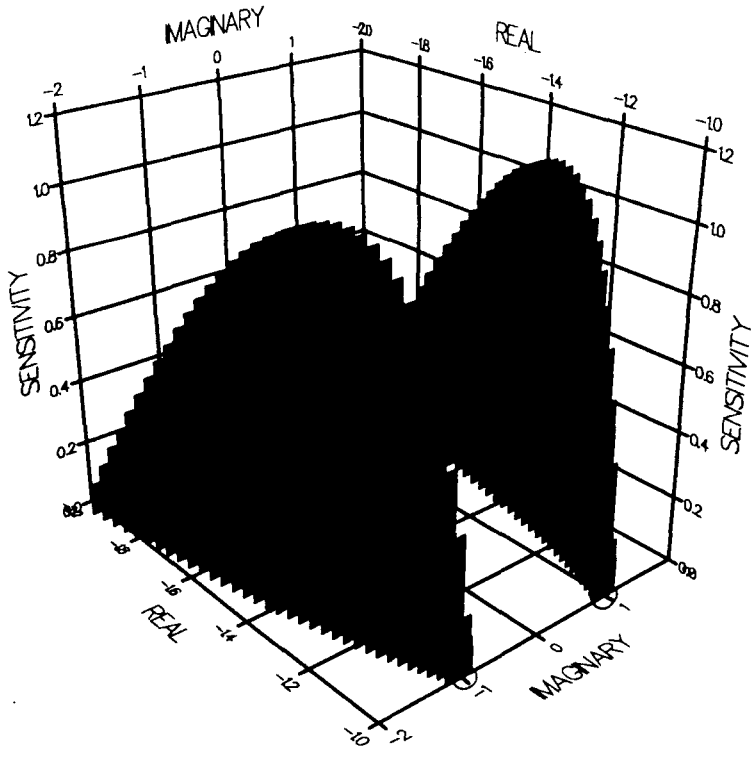
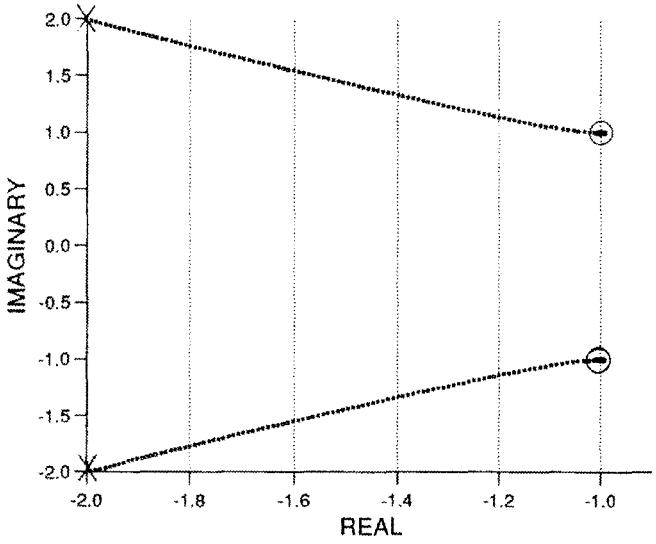
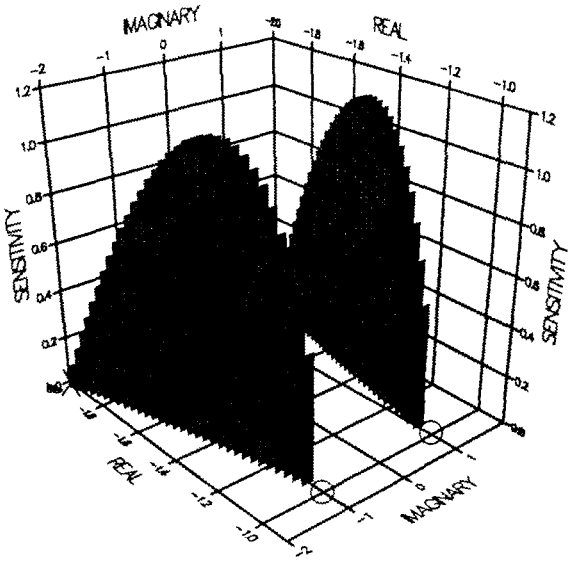


FIGURE 4.2 Y SECOND-ORDER PLANT

ROOT LOCUS

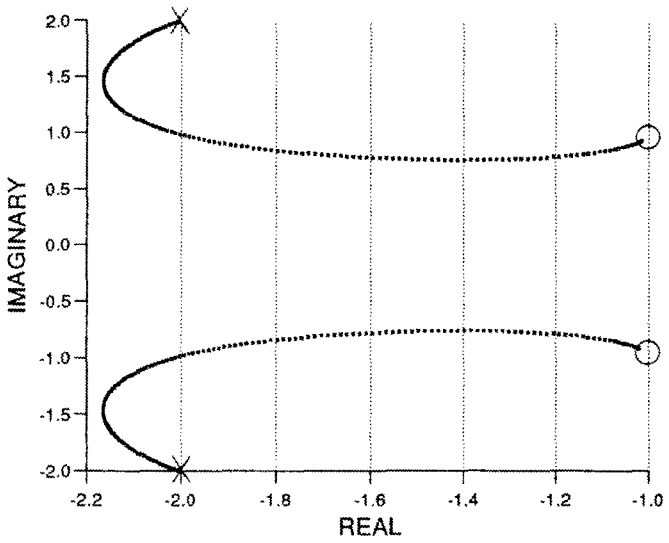


3-D SENSITIVITY PROFILE

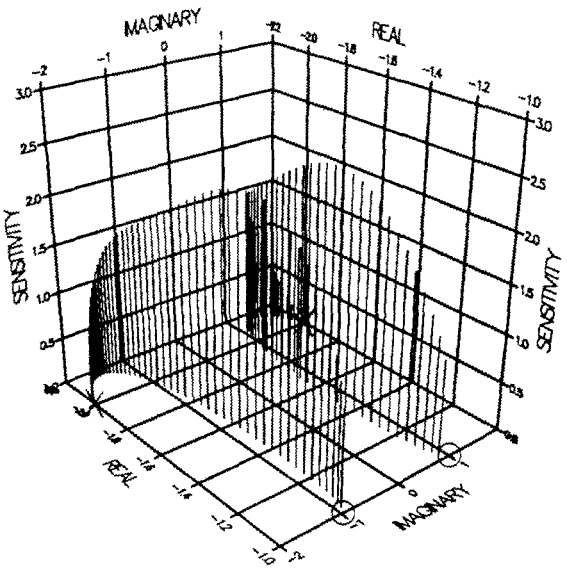


0.2 sec Delay

ROOT LOCUS



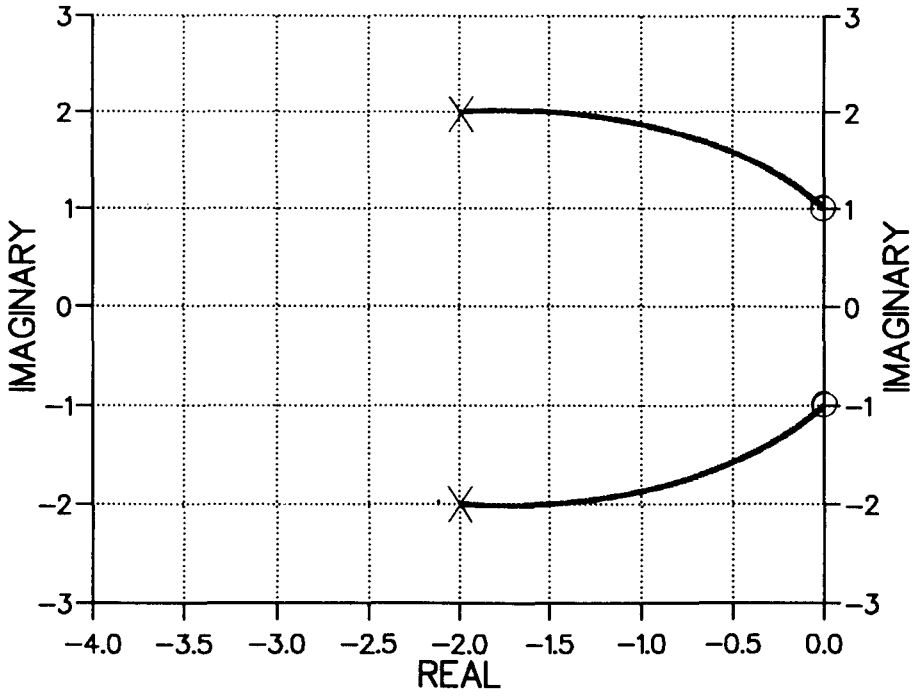
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 y SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

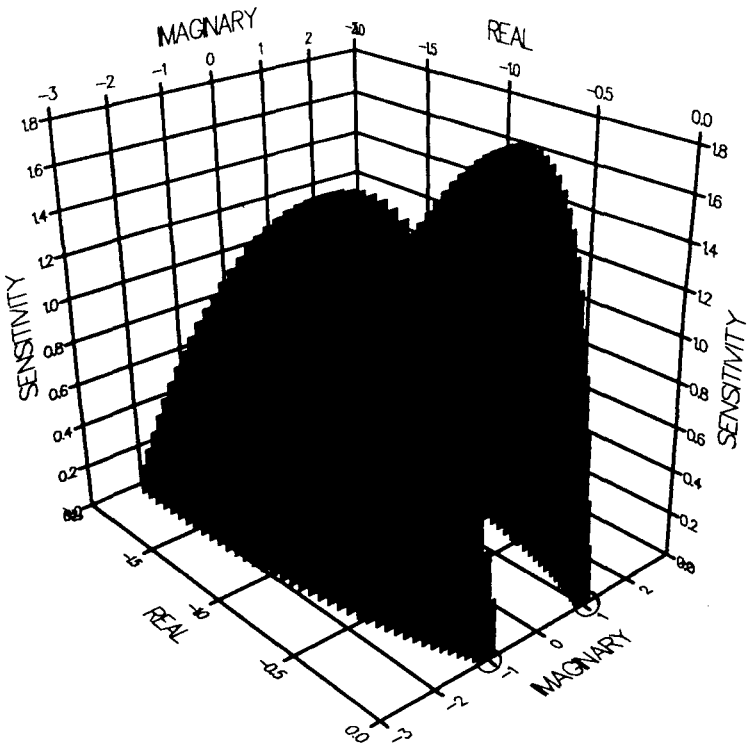
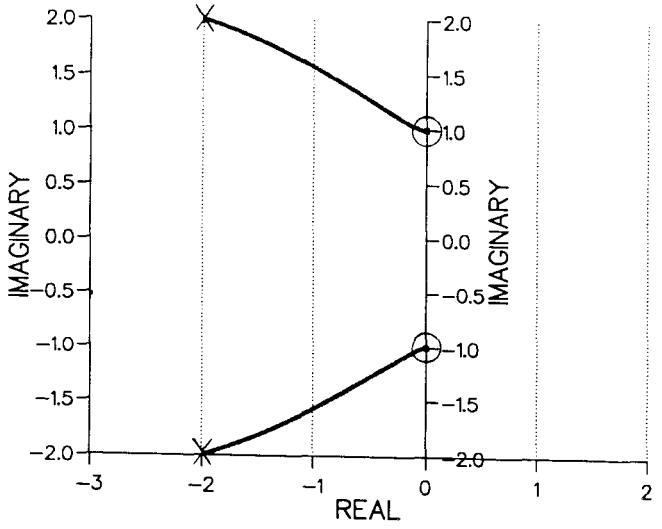
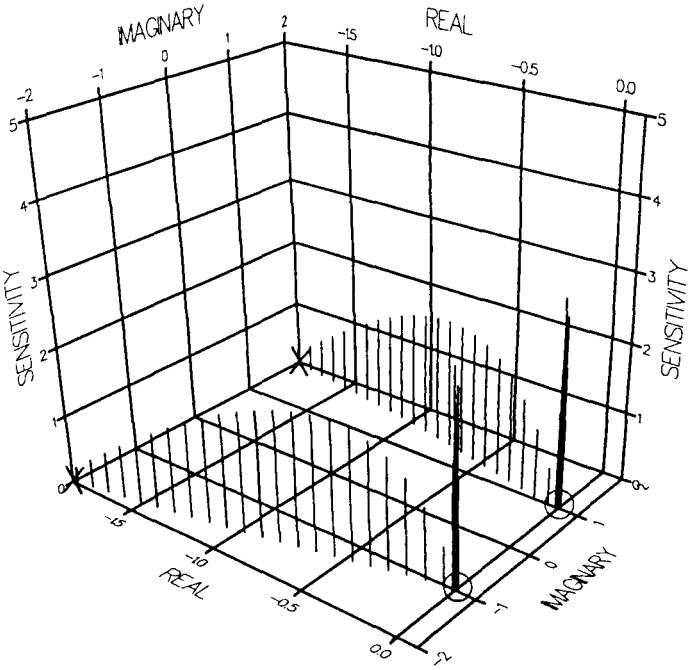


FIGURE 4.2 Z SECOND-ORDER PLANT

ROOT LOCUS

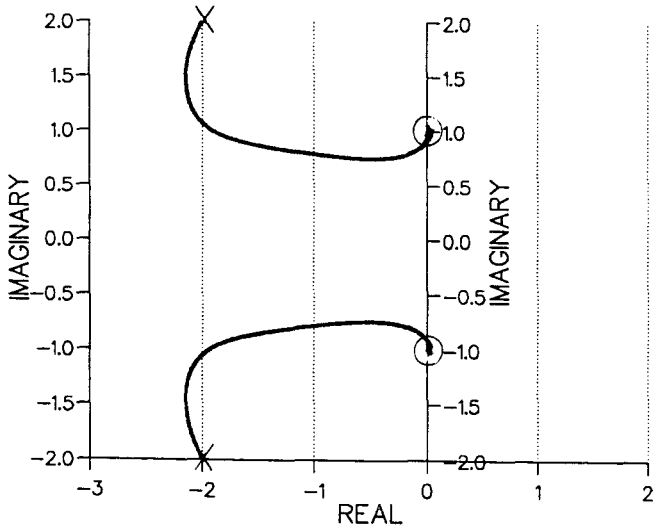


3-D SENSITIVITY PROFILE

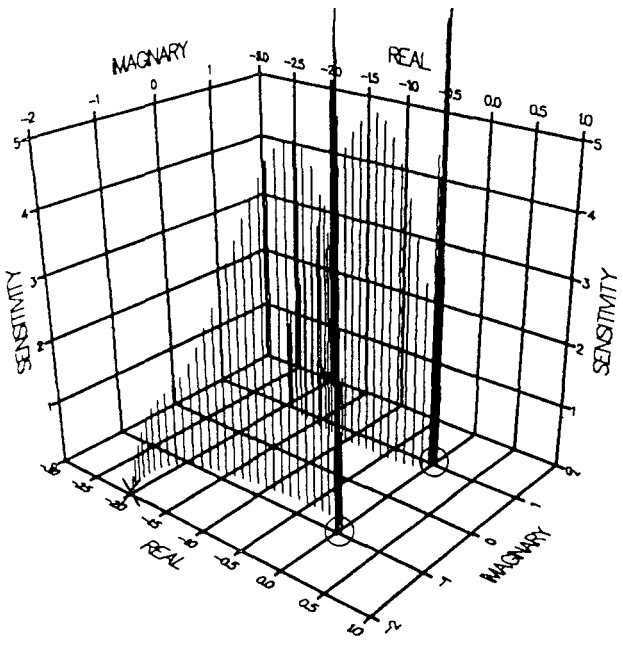


0.2 sec Delay

ROOT LOCUS



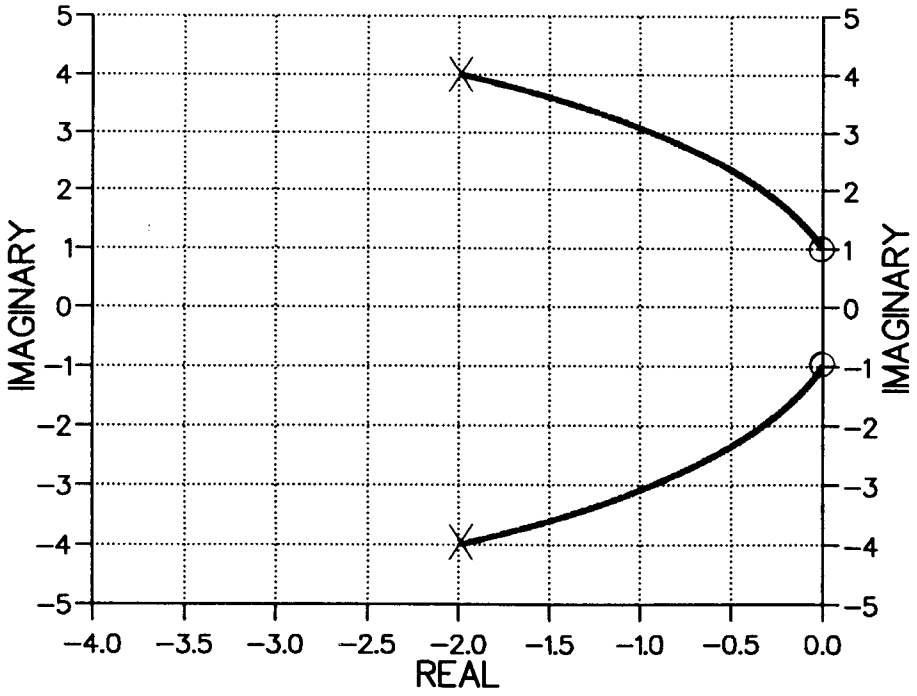
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 z SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

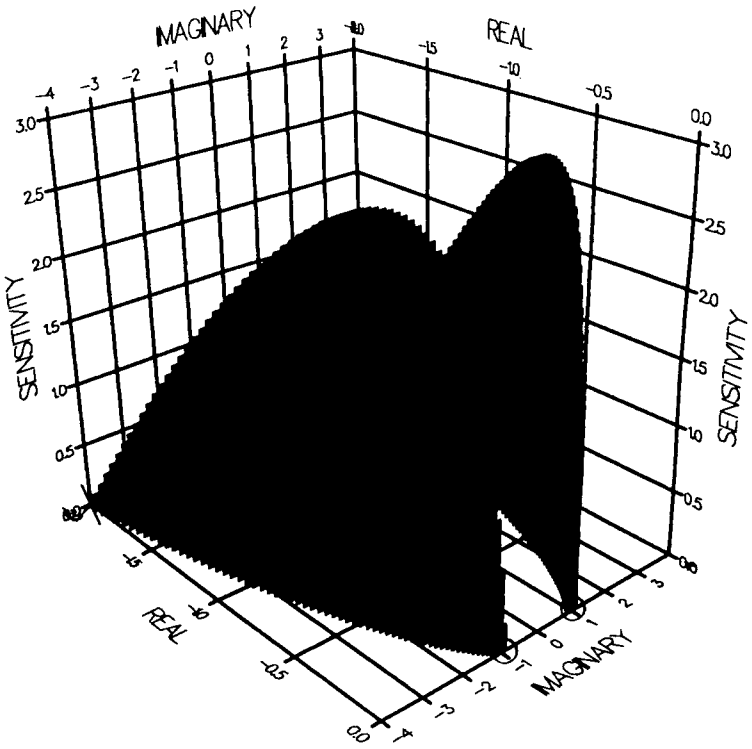
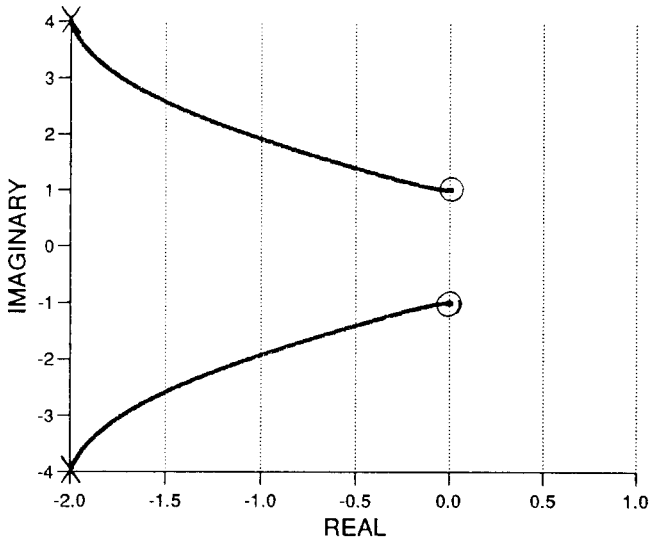
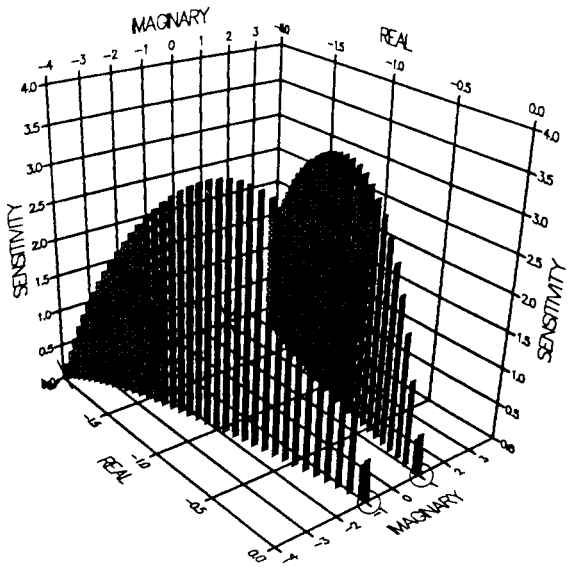


FIGURE 4.2 #A SECOND-ORDER PLANT

ROOT LOCUS

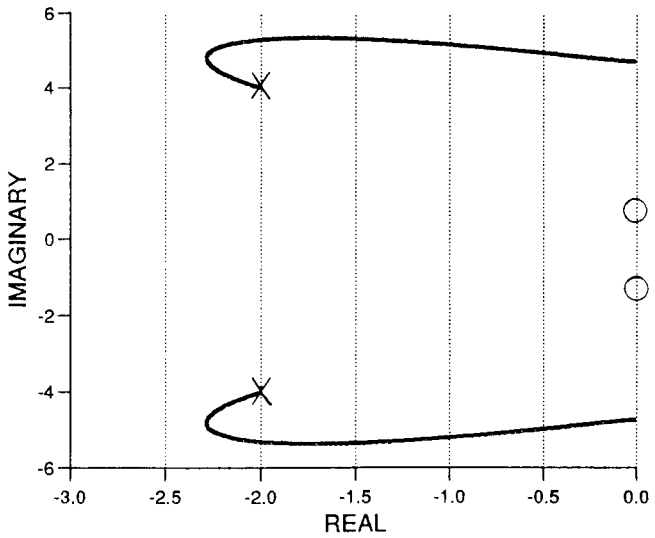


3-D SENSITIVITY PROFILE

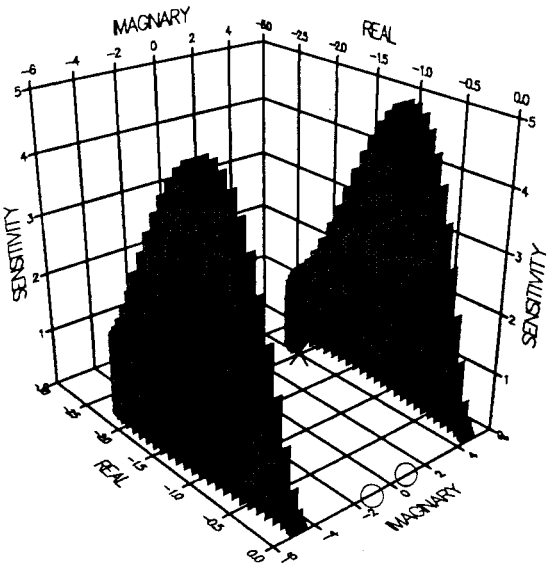


0.2 sec Delay

ROOT LOCUS



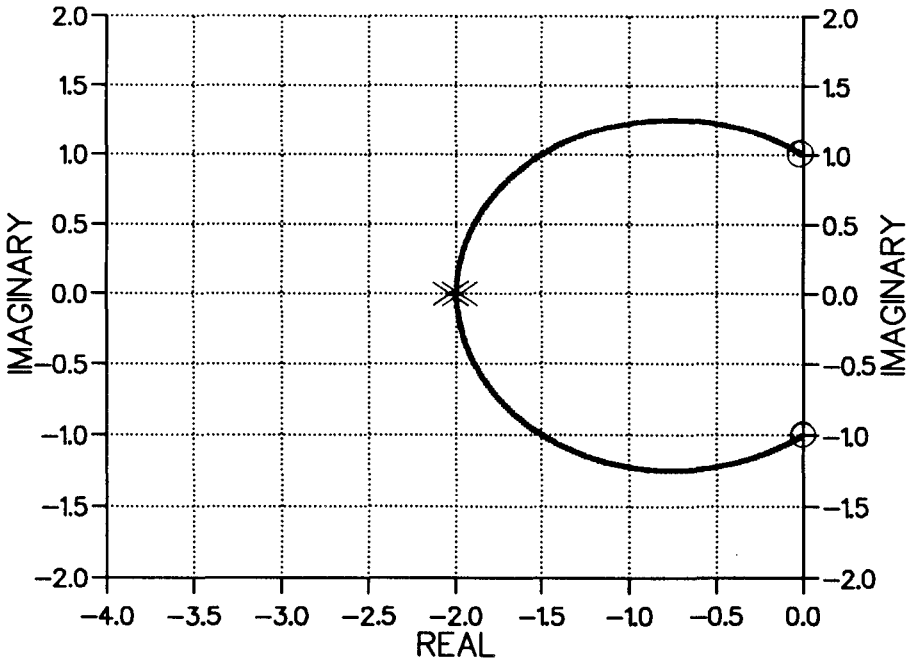
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 #a SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

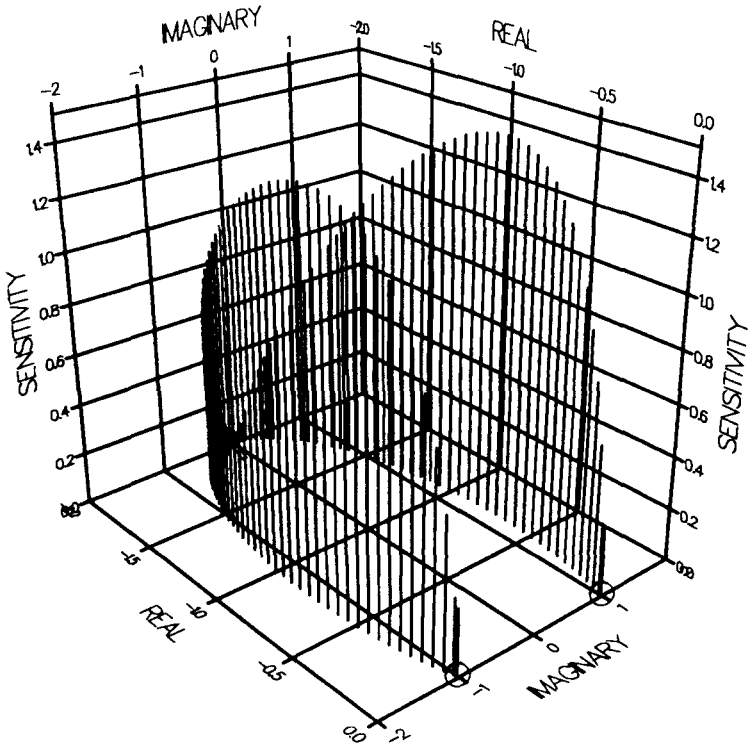
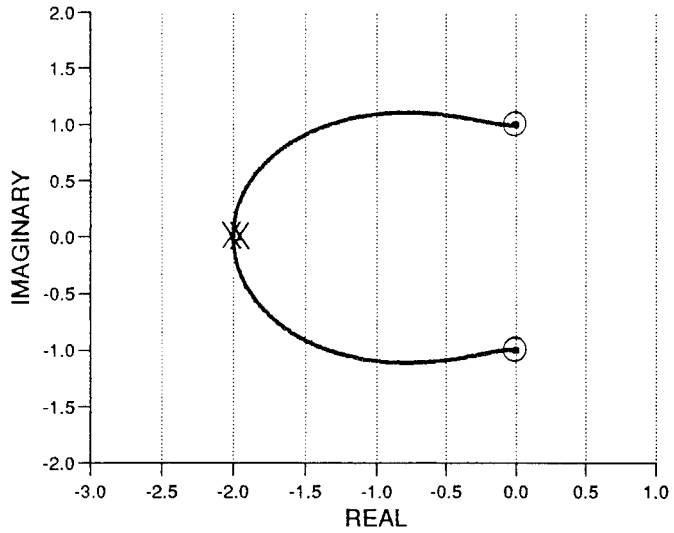
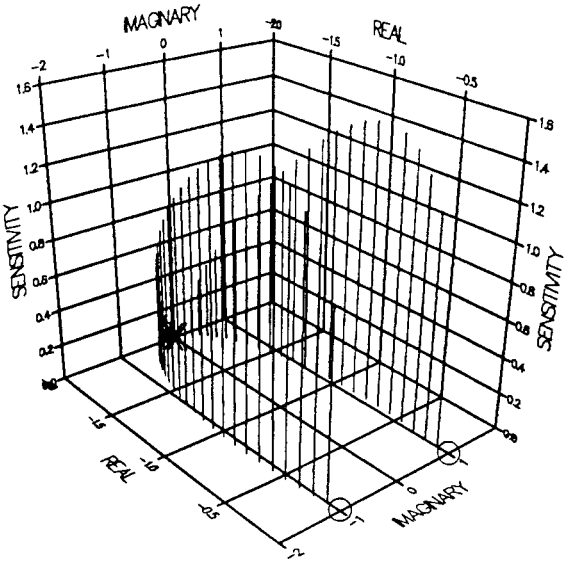


FIGURE 4.2 #B SECOND-ORDER PLANT

ROOT LOCUS

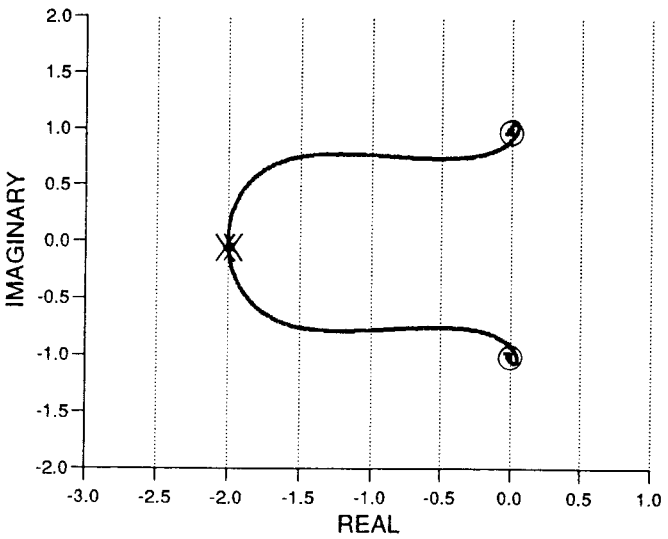


3-D SENSITIVITY PROFILE

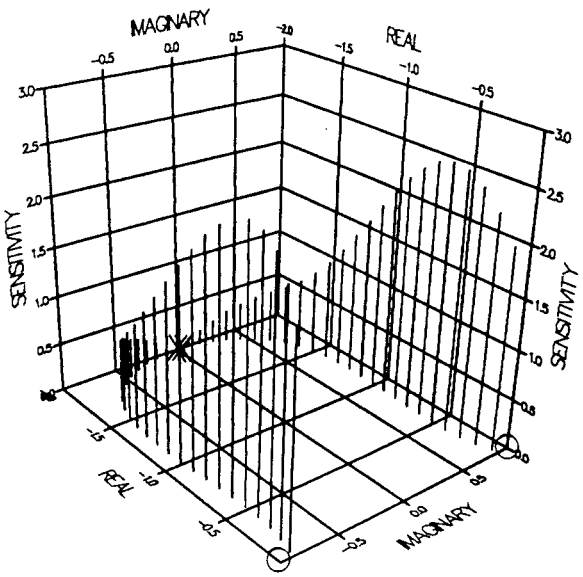


0.2 sec Delay

ROOT LOCUS



3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 #b SECOND-ORDER PLANT WITH DELAYS

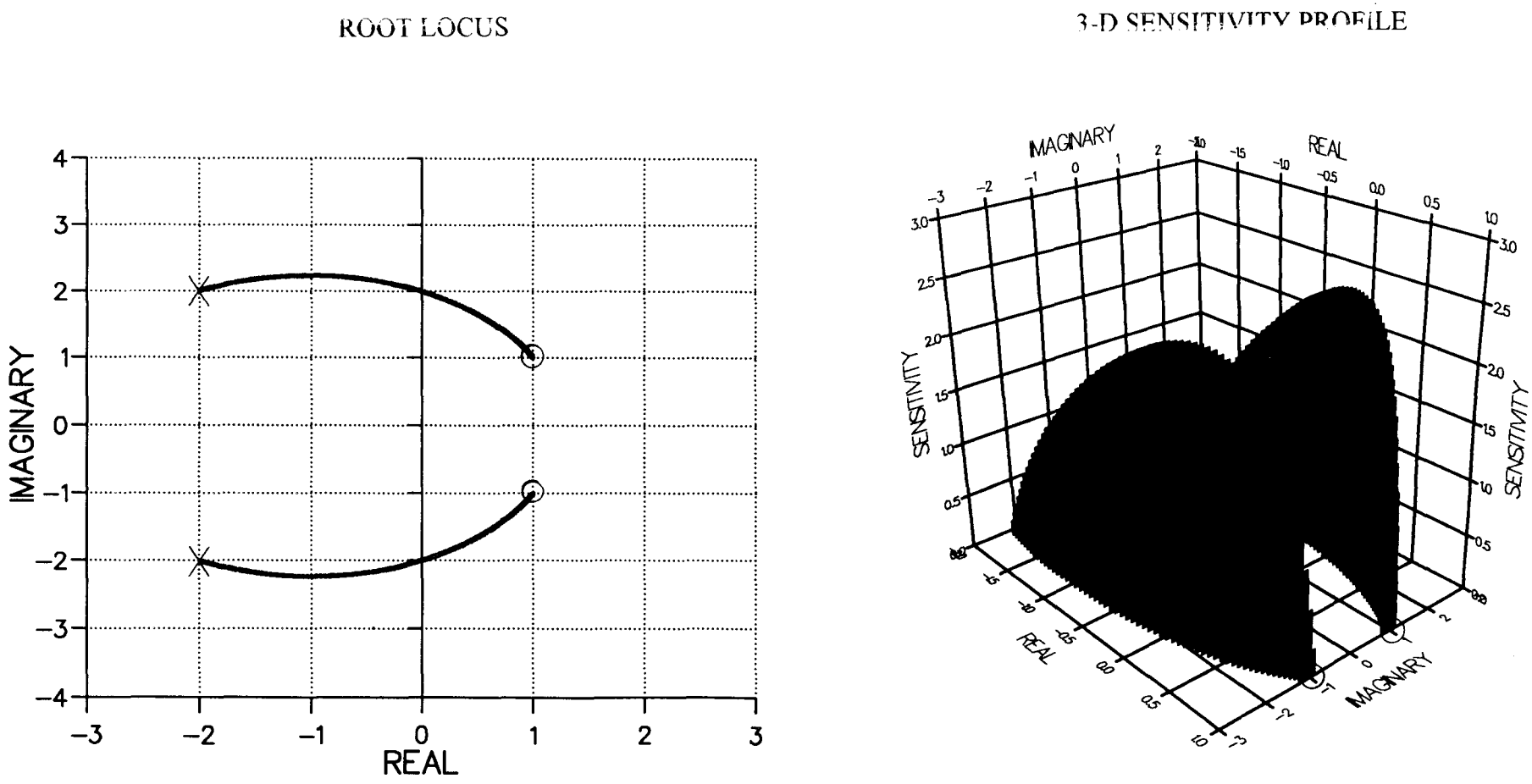
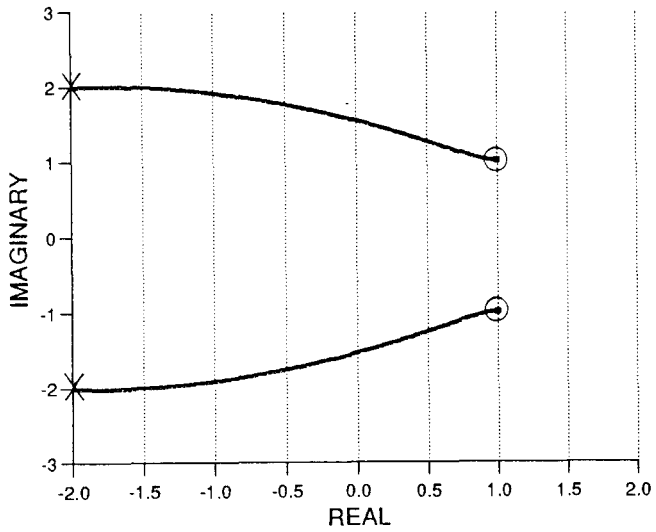
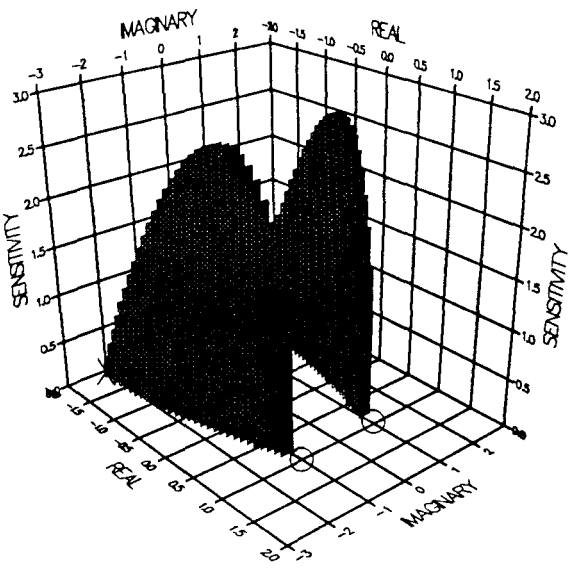


FIGURE 4.2 #C SECOND-ORDER PLANT

ROOT LOCUS

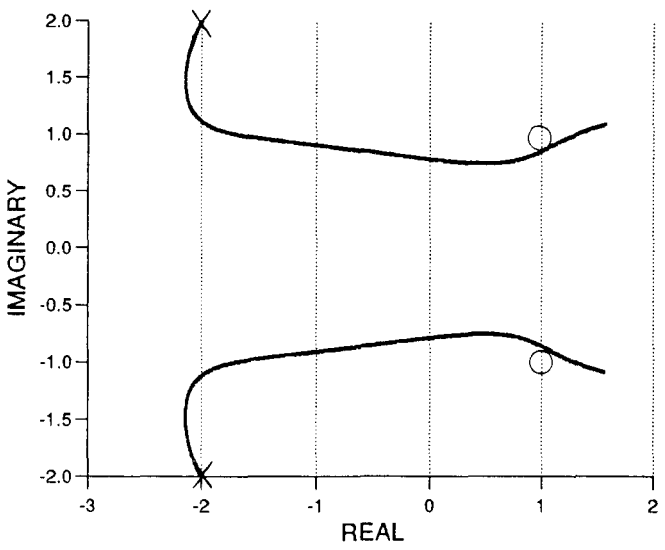


3-D SENSITIVITY PROFILE

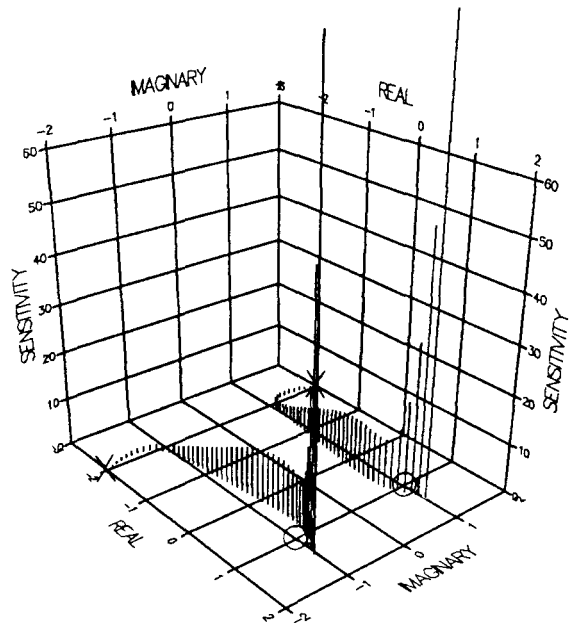


0.2 sec Delay

ROOT LOCUS



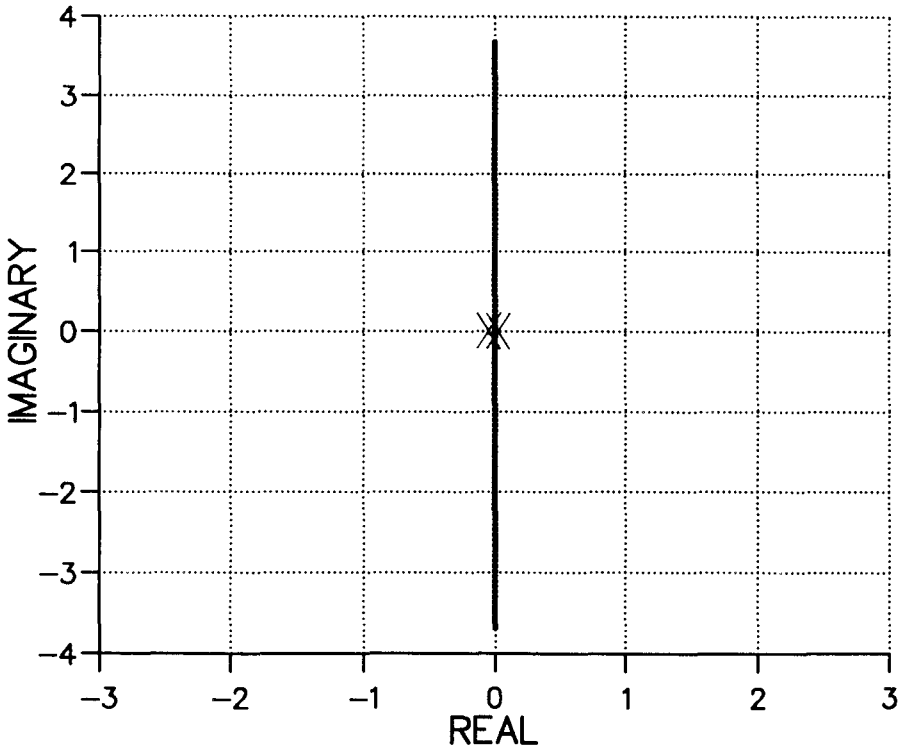
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 #c SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

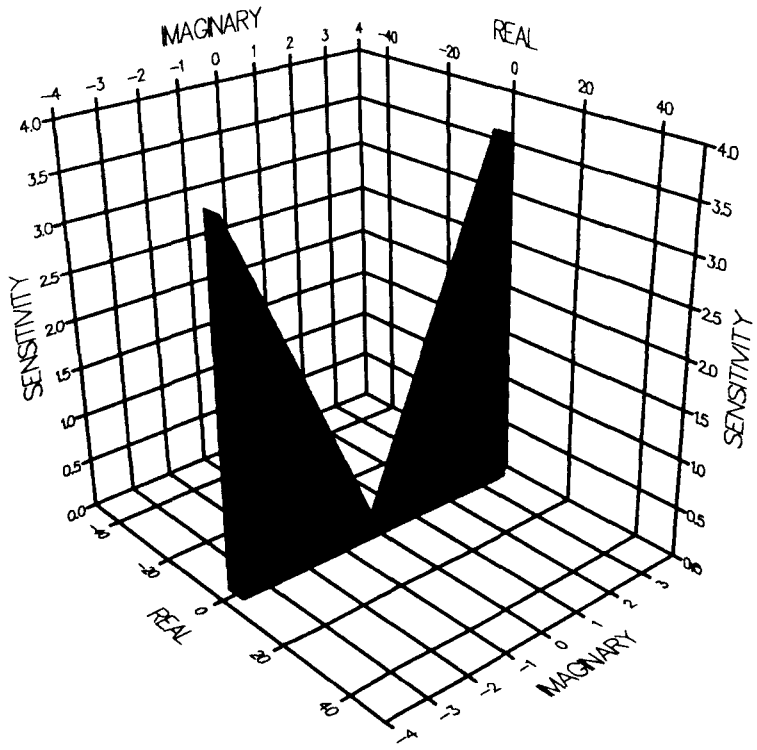
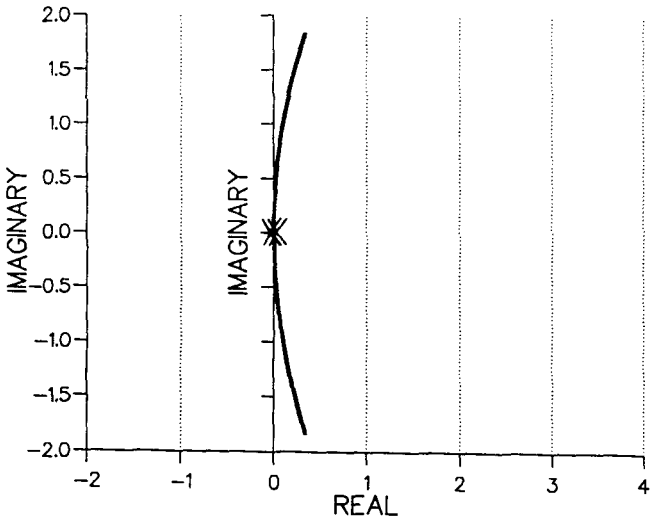
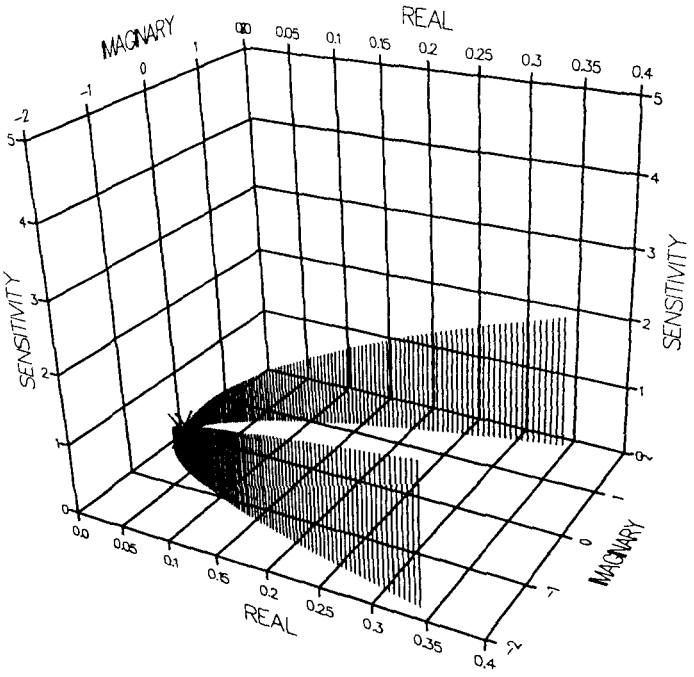


FIGURE 4.2 #D SECOND-ORDER PLANT

ROOT LOCUS

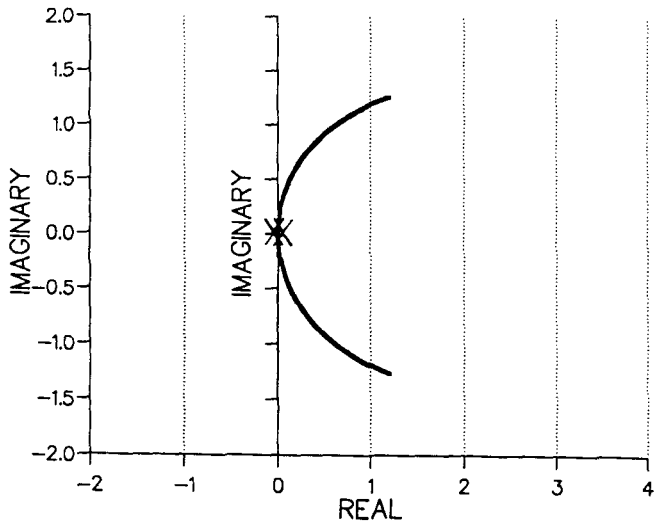


3-D SENSITIVITY PROFILE

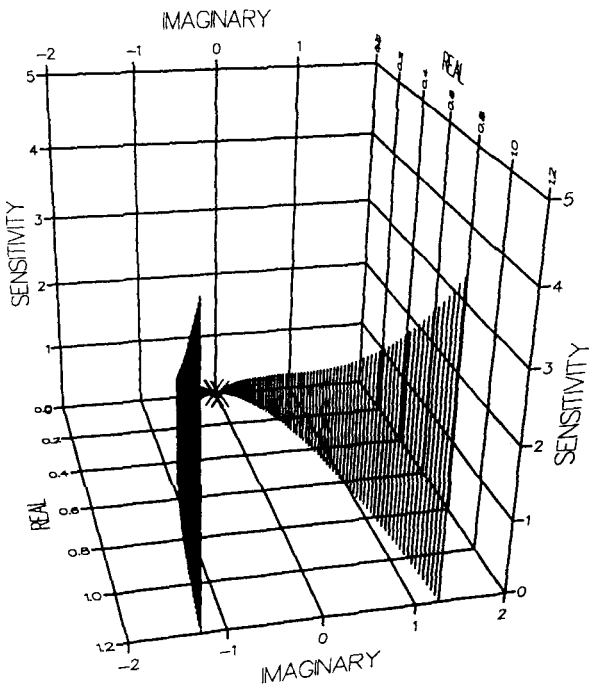


0.2 sec Delay

ROOT LOCUS



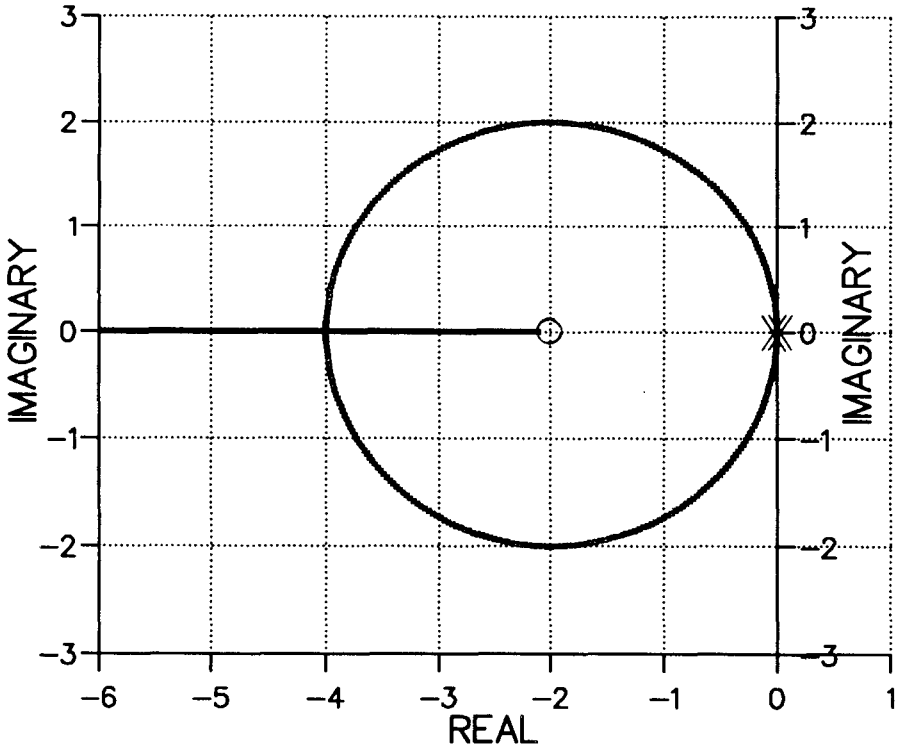
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 #d SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

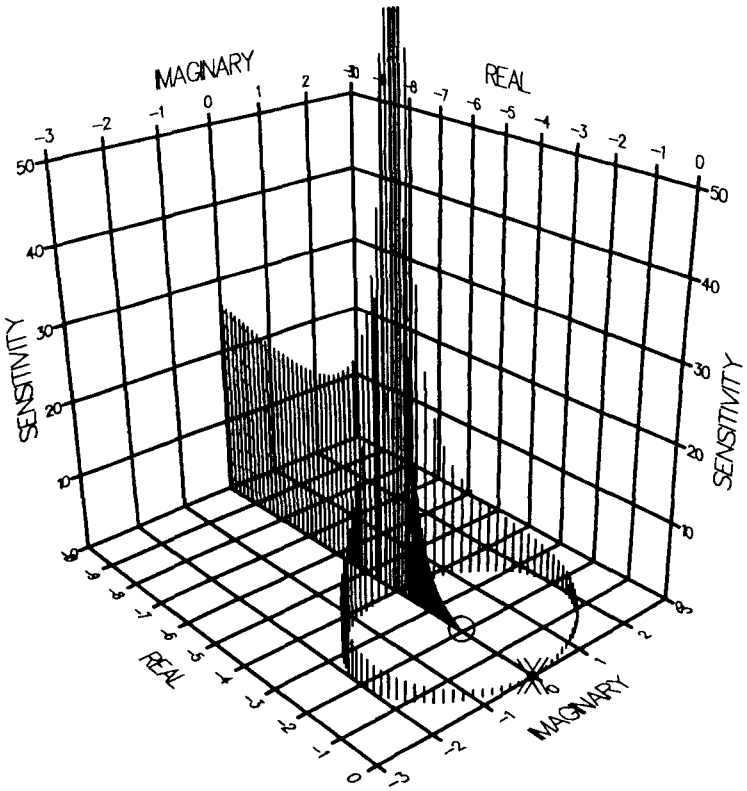
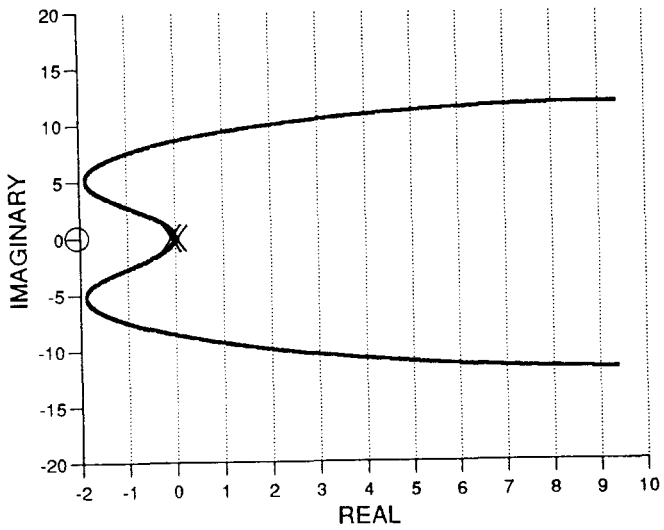
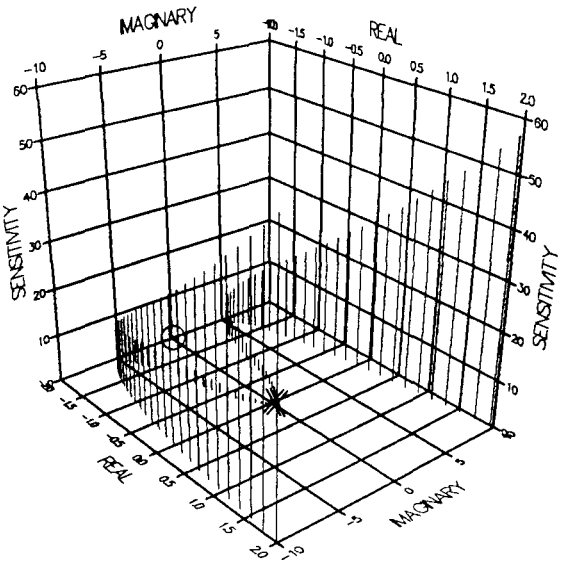


FIGURE 4.2 #E SECOND-ORDER PLANT

ROOT LOCUS

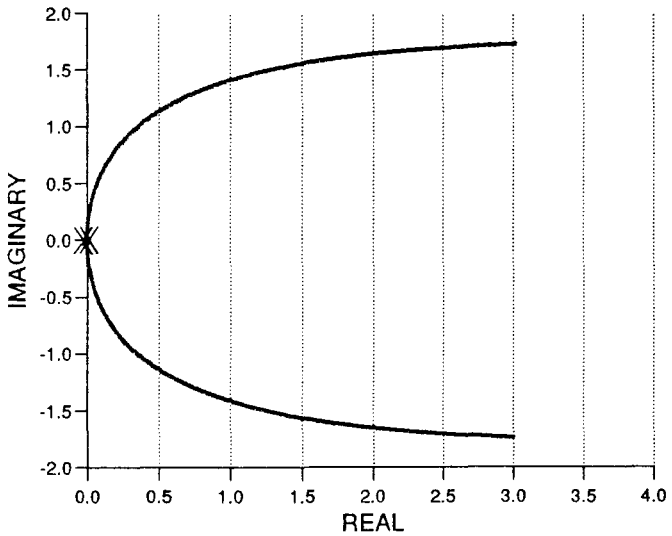


3-D SENSITIVITY PROFILE

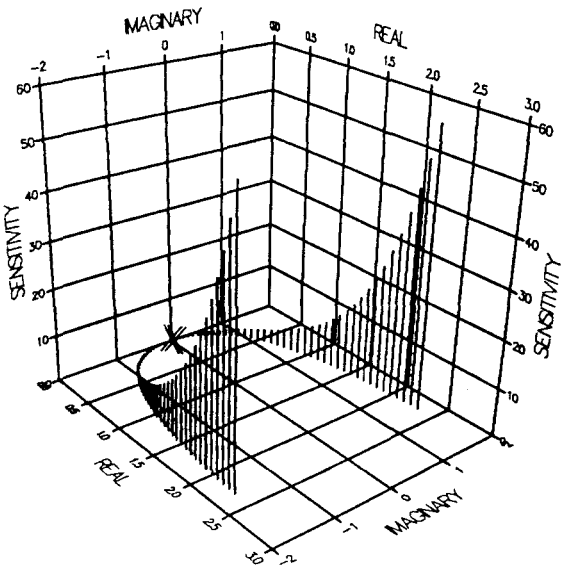


0.2 sec Delay

ROOT LOCUS



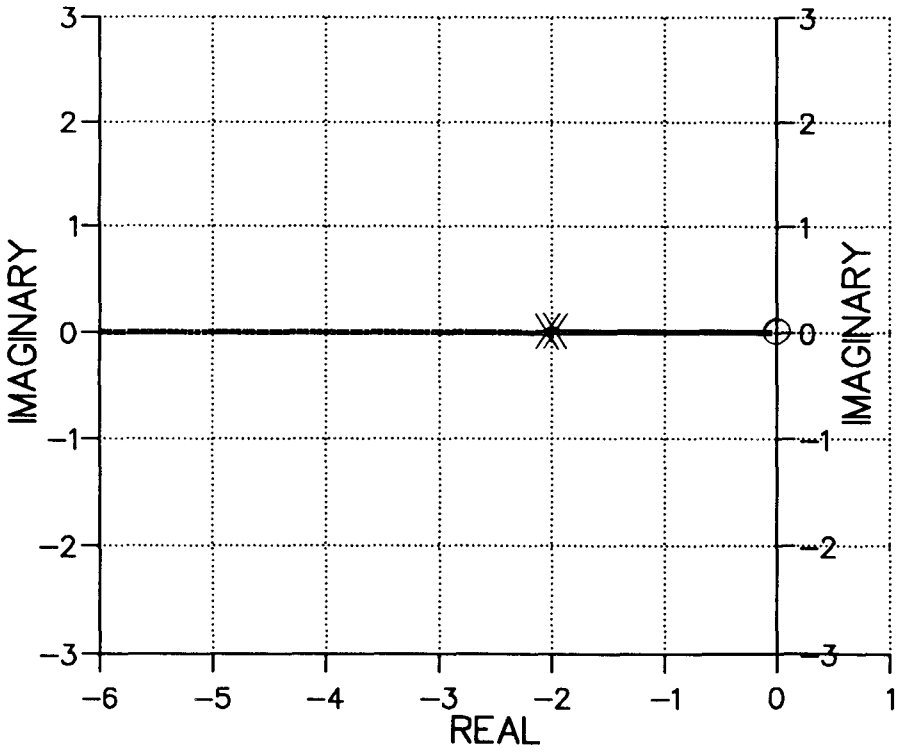
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 #e SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

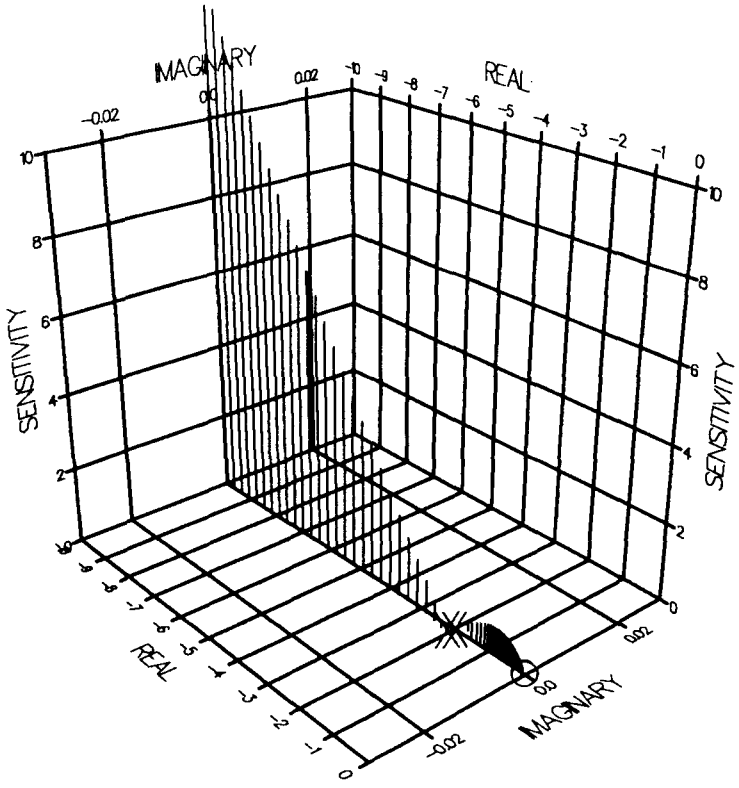
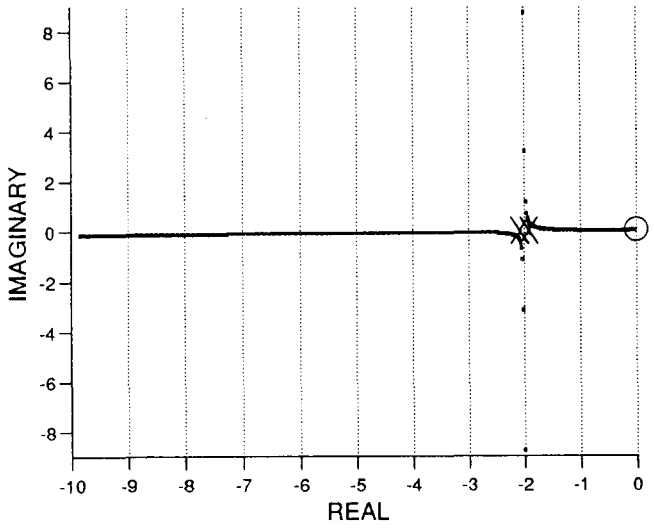
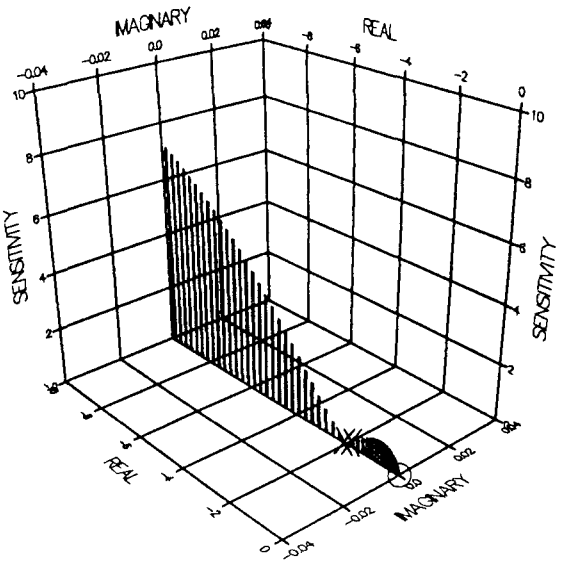


FIGURE 4.2 #F SECOND-ORDER PLANT

ROOT LOCUS

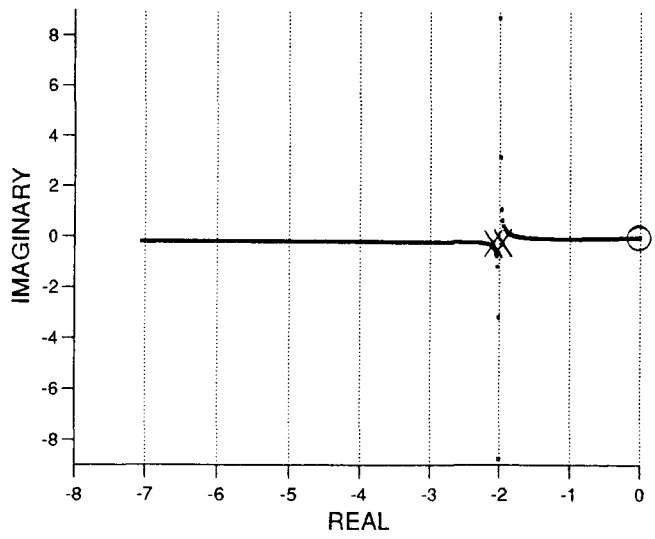


3-D SENSITIVITY PROFILE

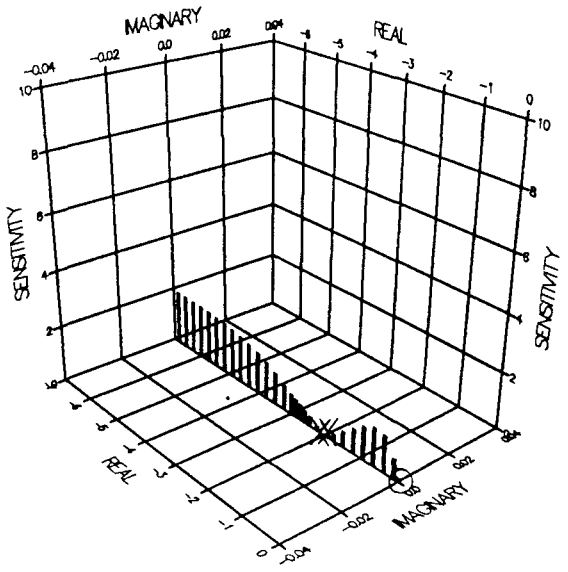


0.2 sec Delay

ROOT LOCUS



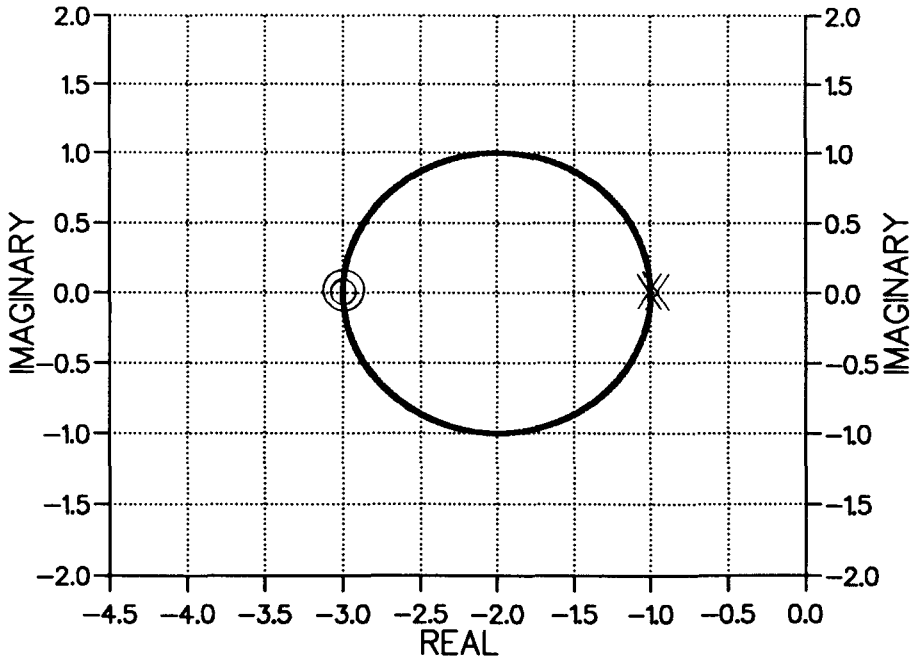
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 #f SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

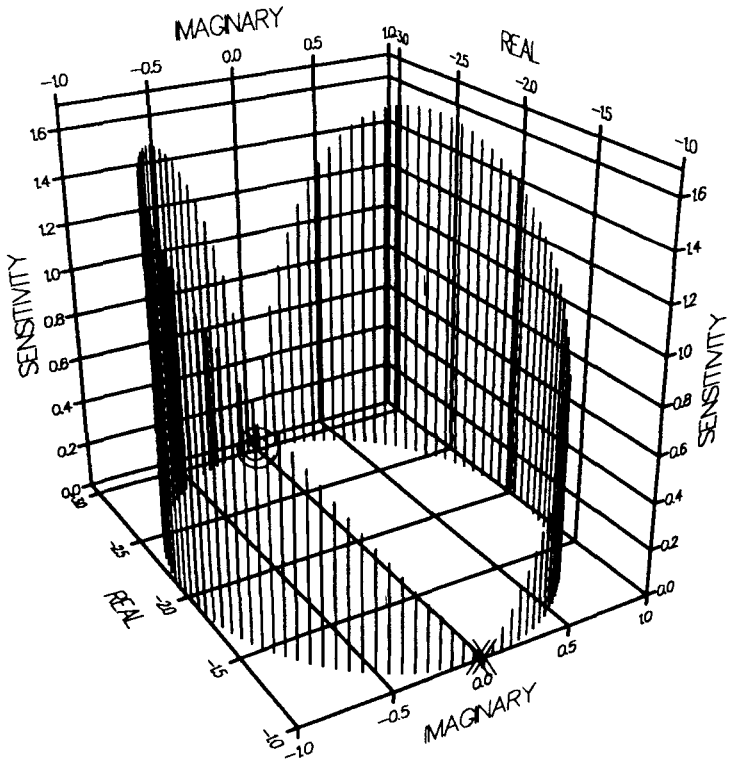
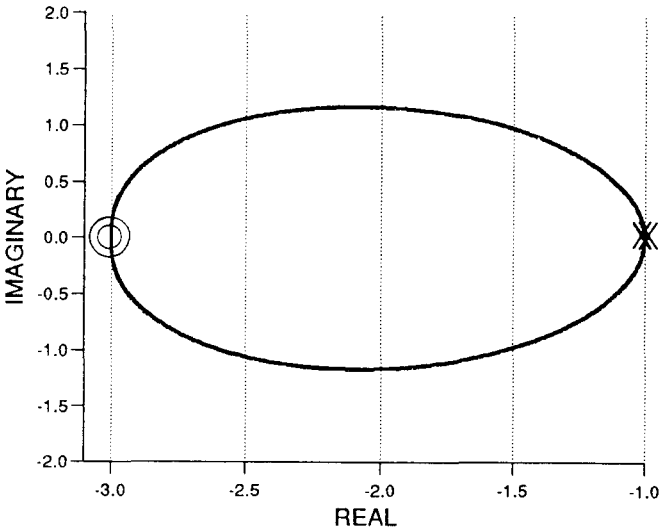
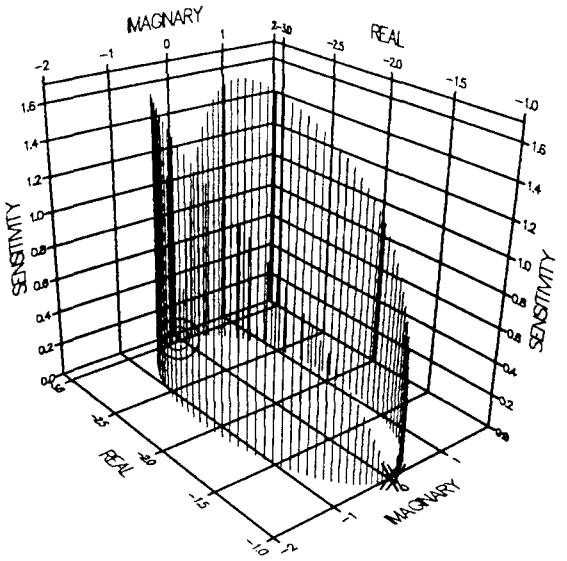


FIGURE 4.2 #G SECOND-ORDER PLANT

ROOT LOCUS

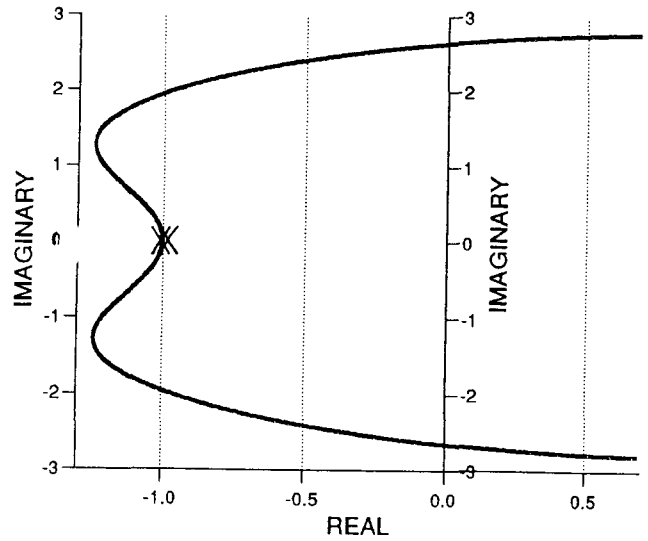


3-D SENSITIVITY PROFILE

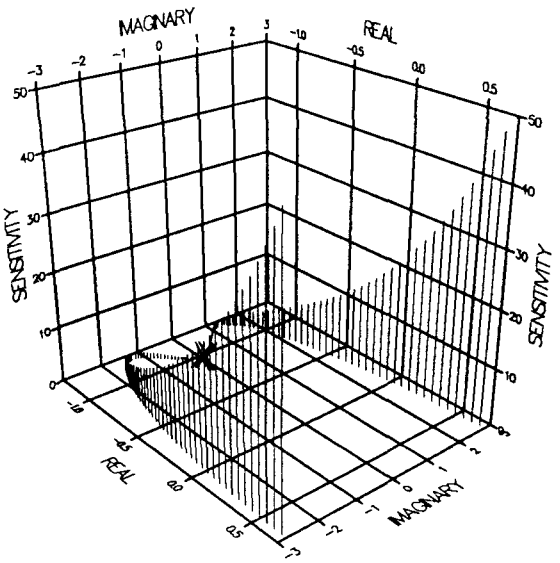


0.2 sec Delay

ROOT LOCUS



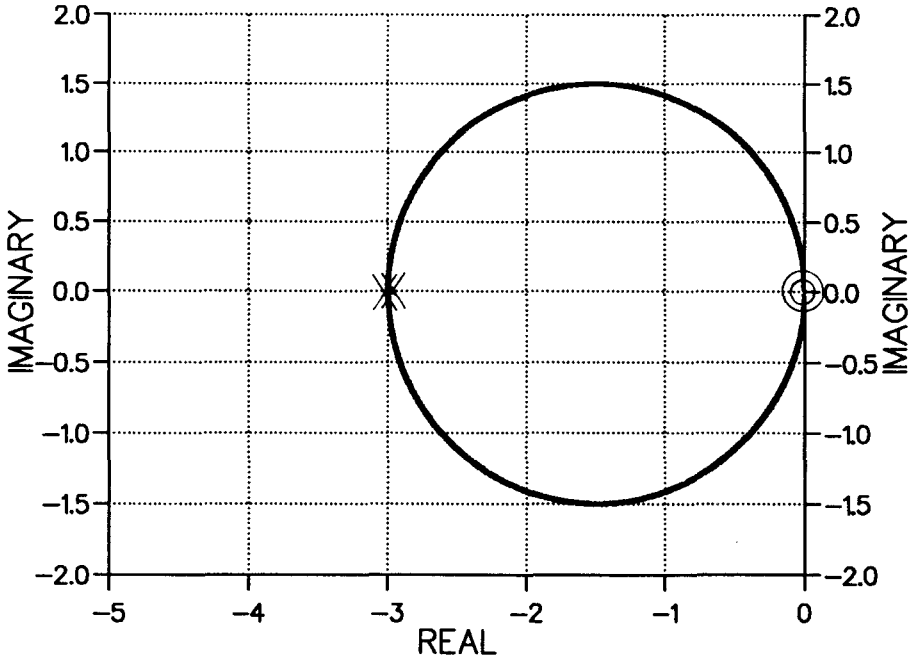
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 #g SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

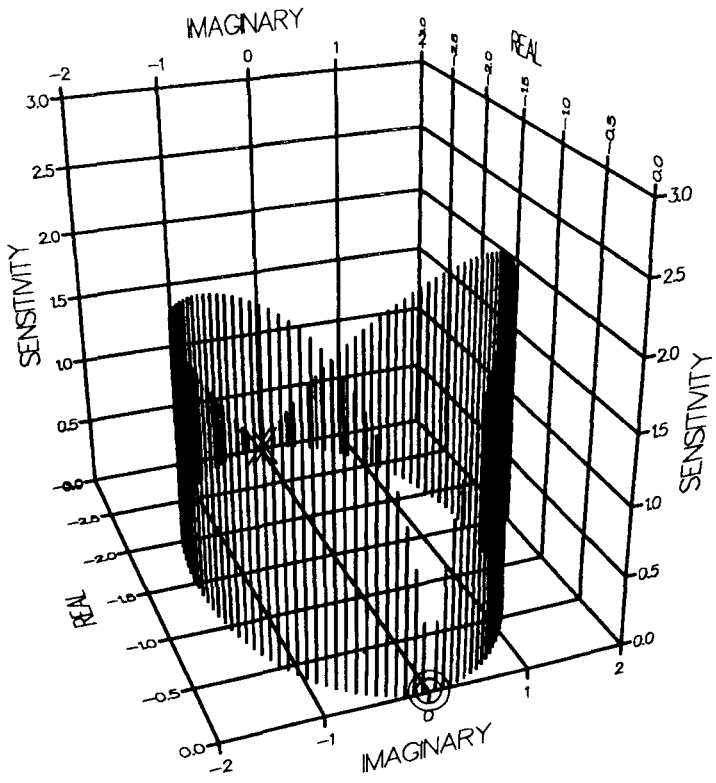
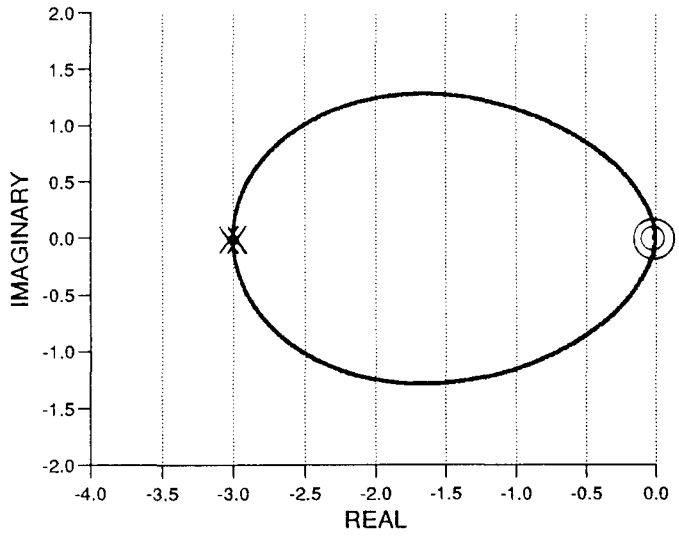
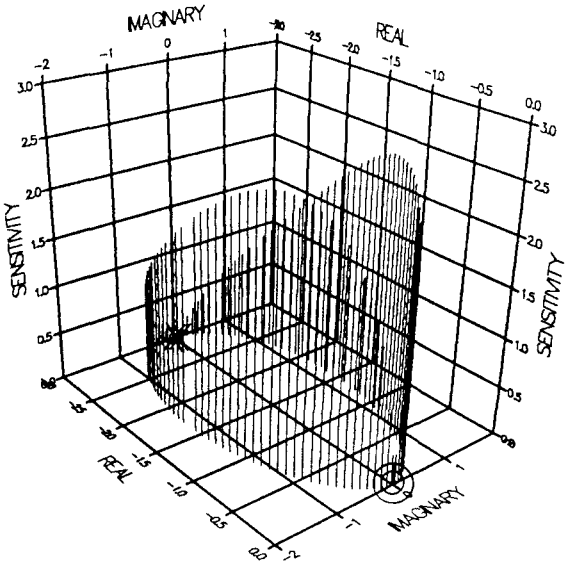


FIGURE 4.2 #H SECOND-ORDER PLANT

ROOT LOCUS

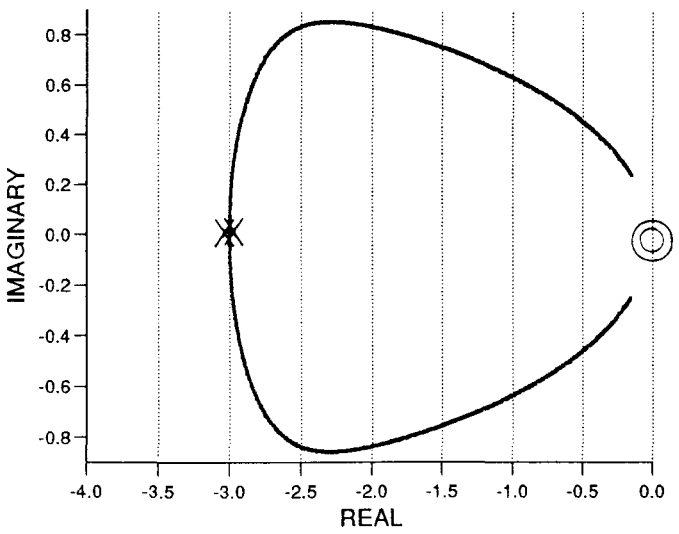


3-D SENSITIVITY PROFILE

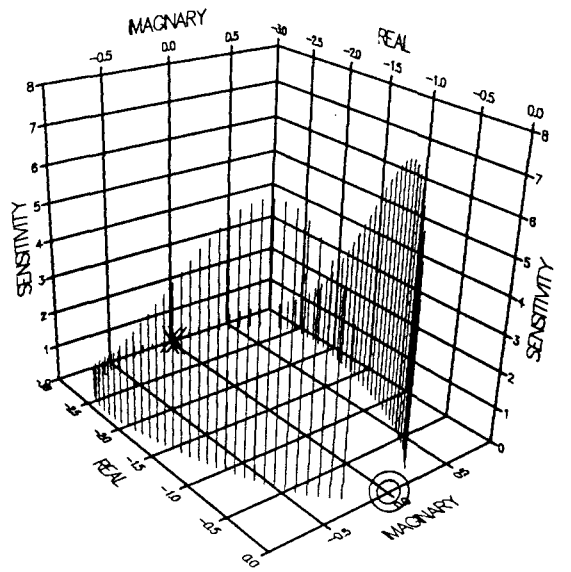


0.2 sec Delay

ROOT LOCUS



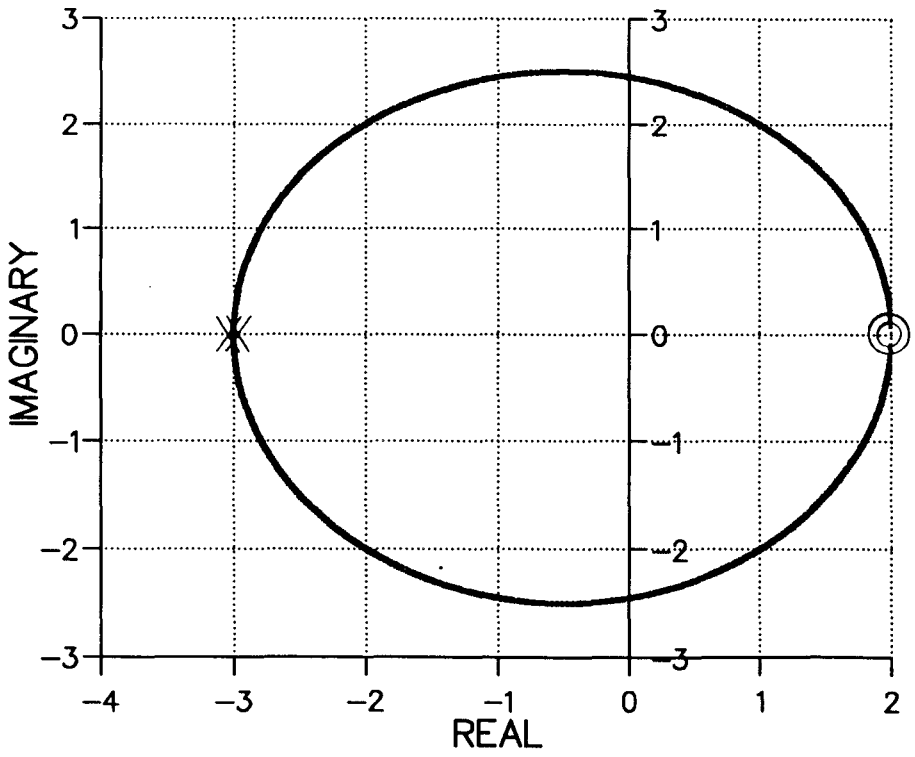
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 #h SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

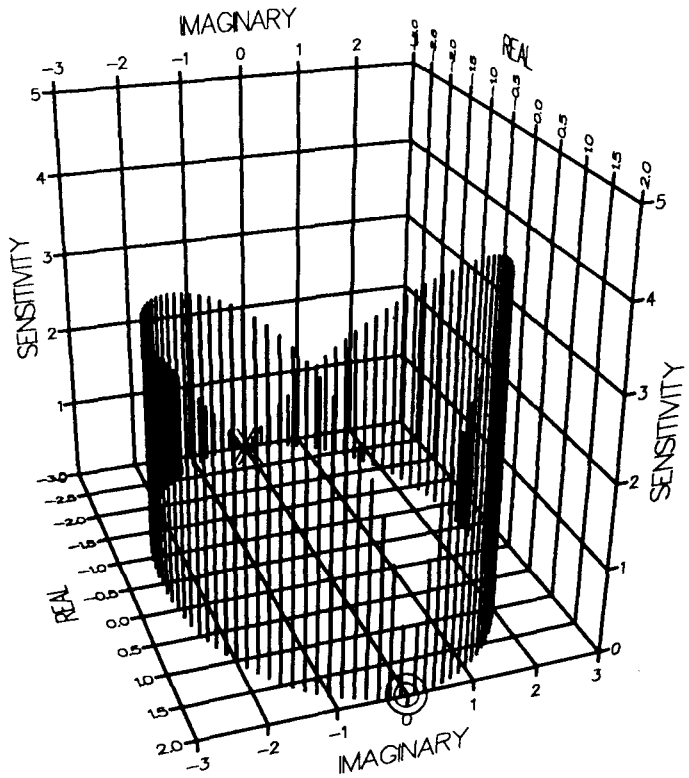
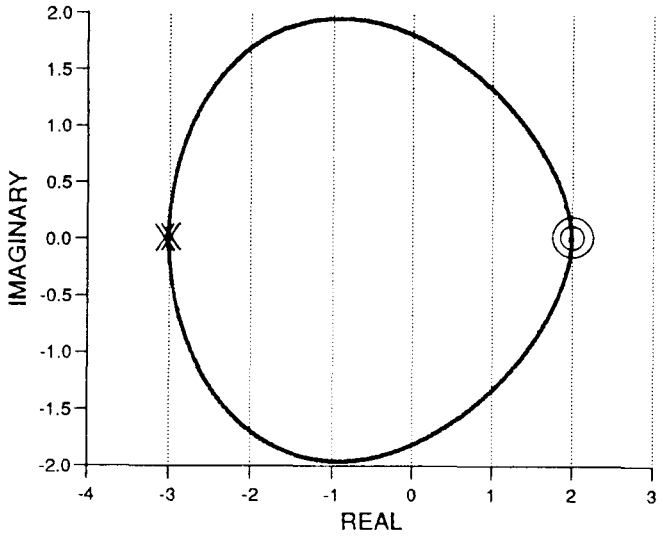
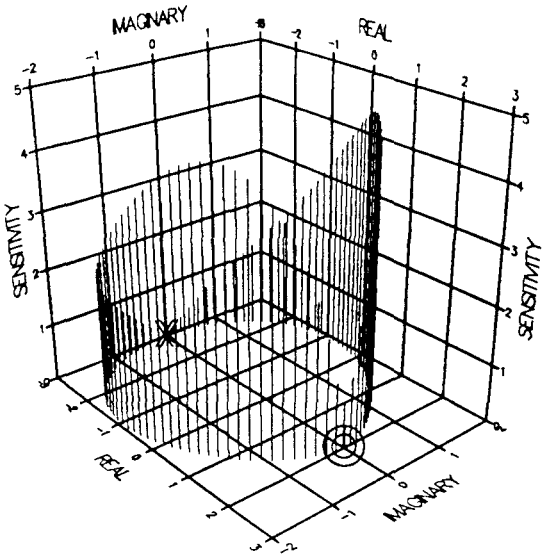


FIGURE 4.2 #I SECOND-ORDER PLANT

ROOT LOCUS

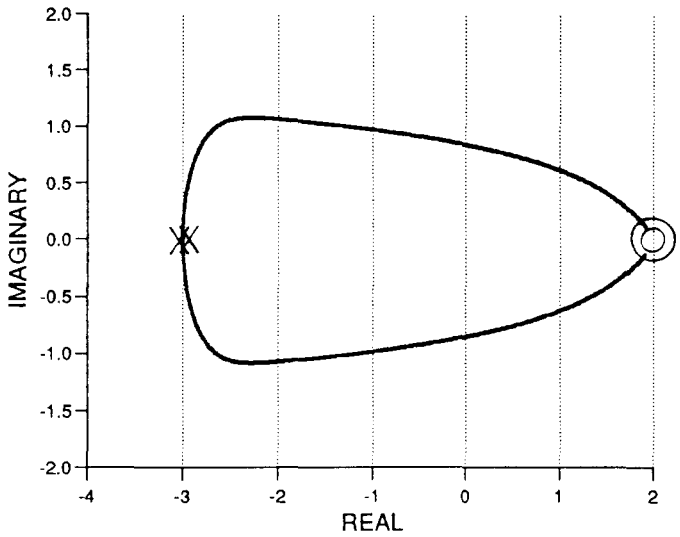


3-D SENSITIVITY PROFILE

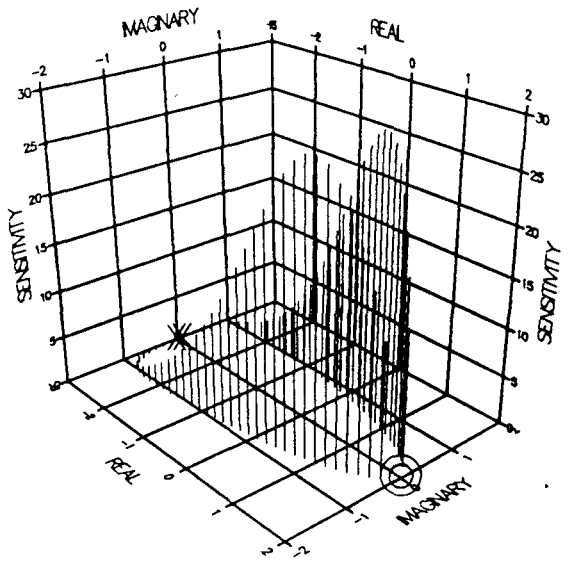


0.2 sec Delay

ROOT LOCUS



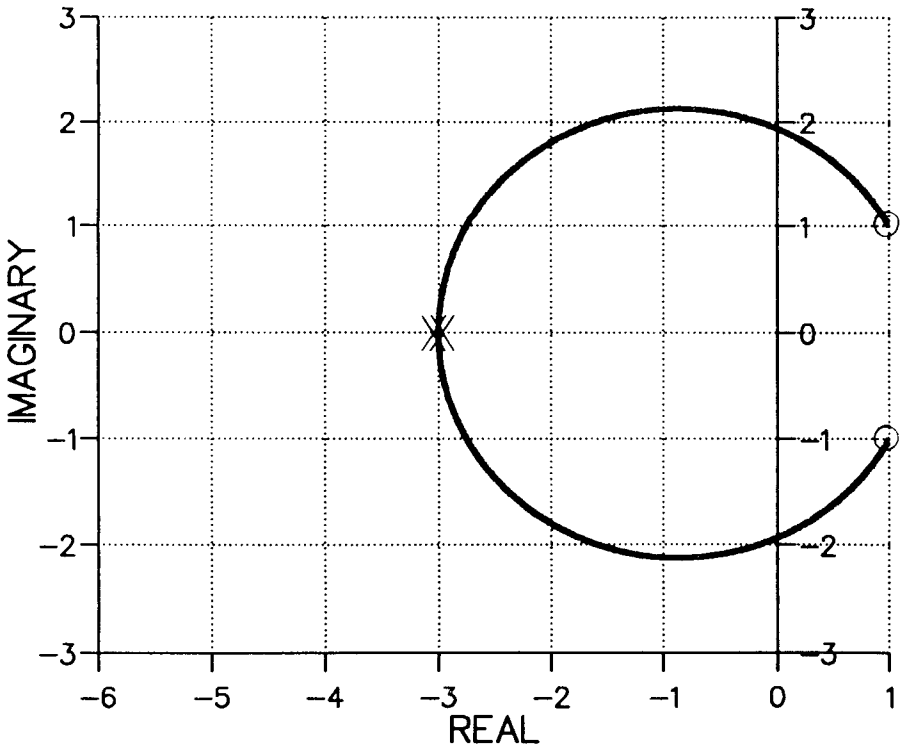
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 #i SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

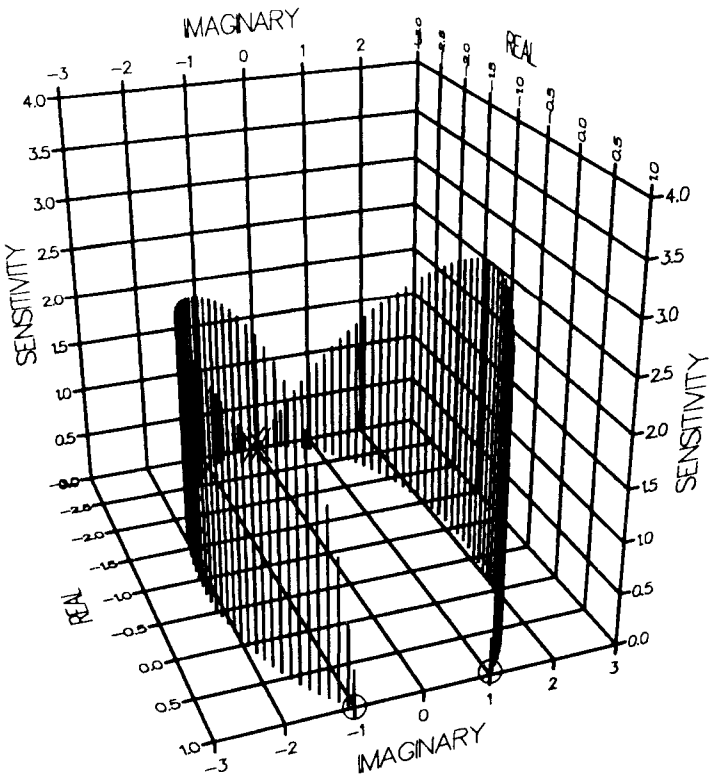
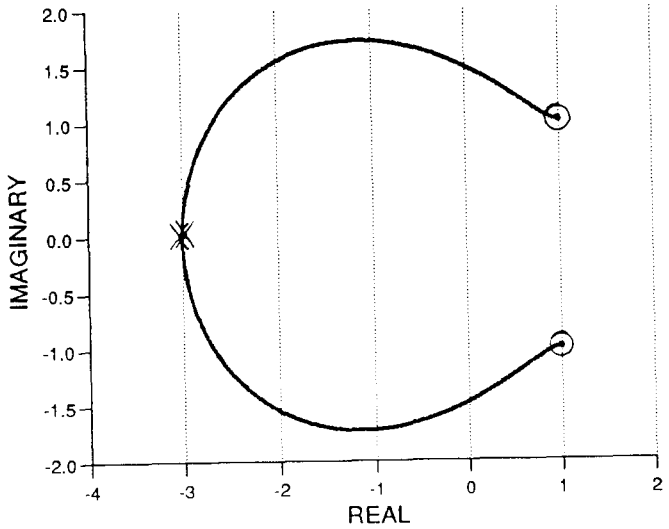
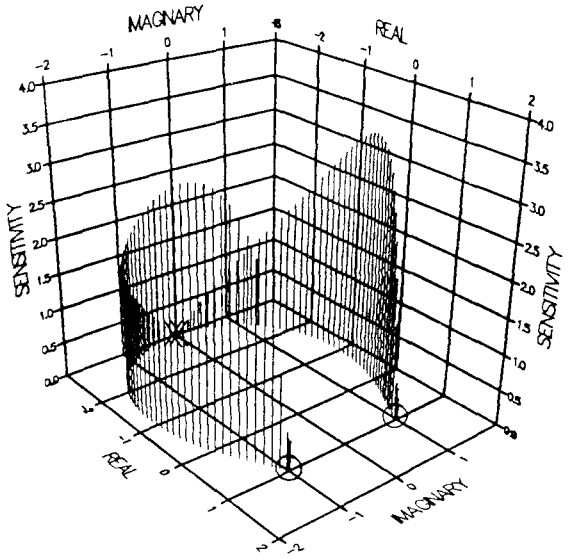


FIGURE 4.2 #J SECOND-ORDER PLANT

ROOT LOCUS

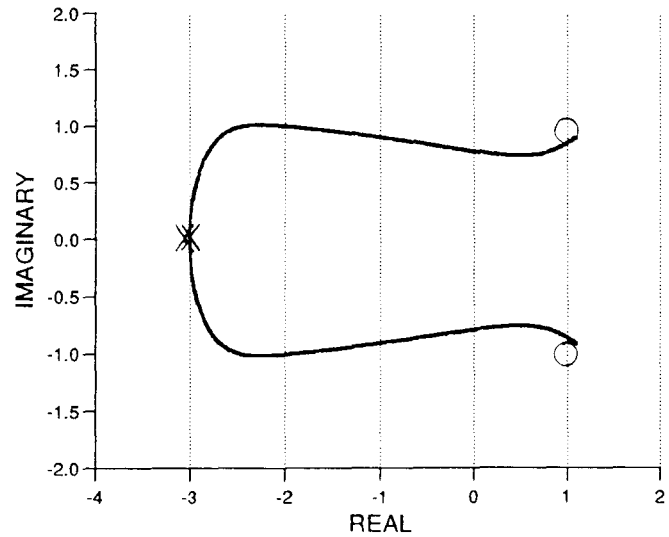


3-D SENSITIVITY PROFILE

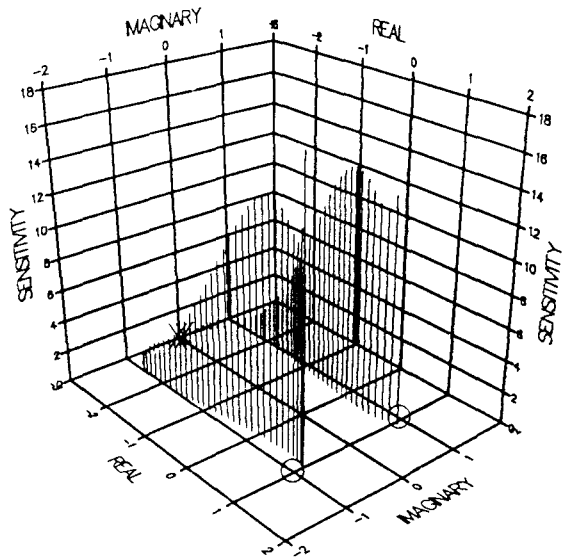


0.2 sec Delay

ROOT LOCUS



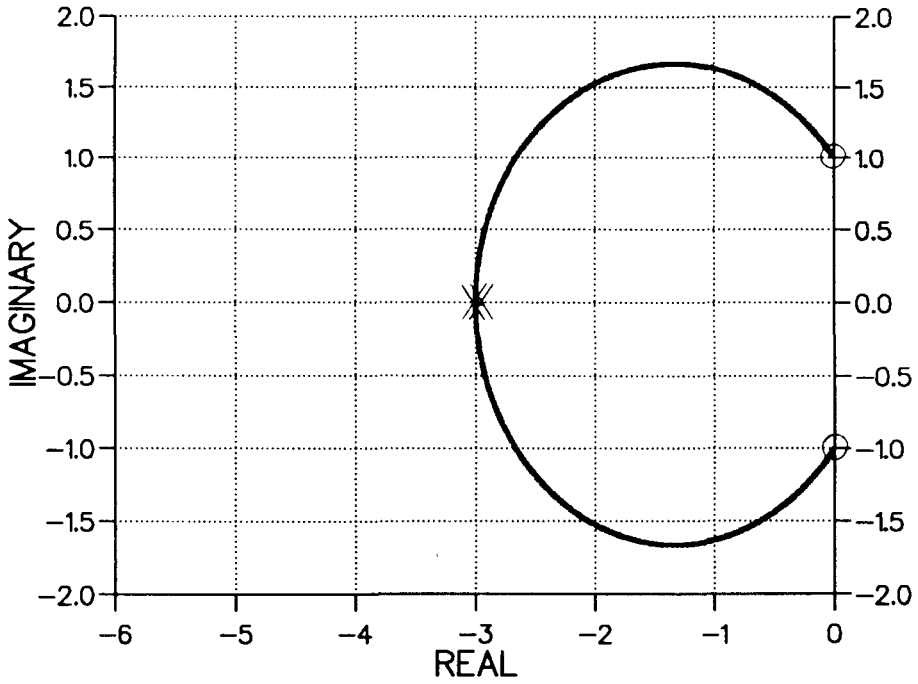
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 #j SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

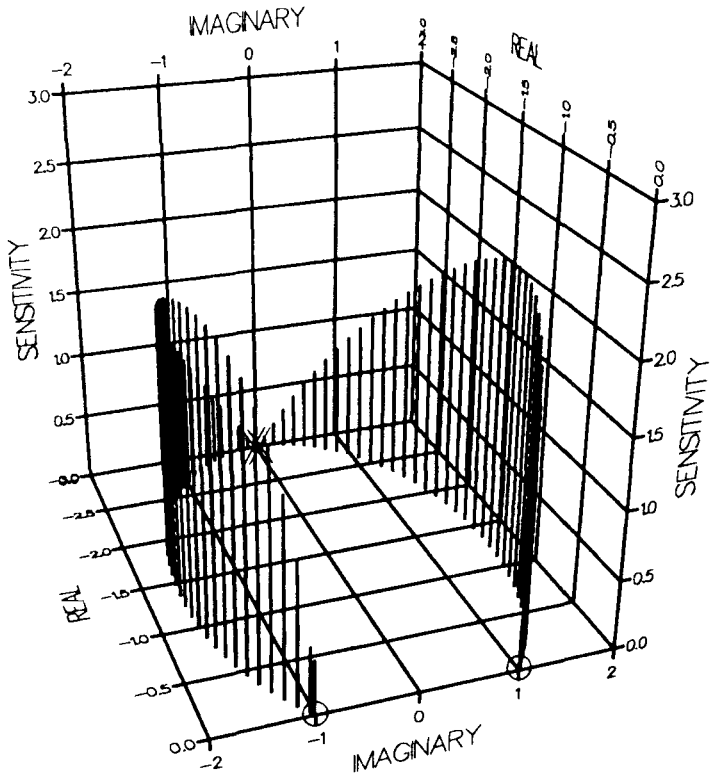
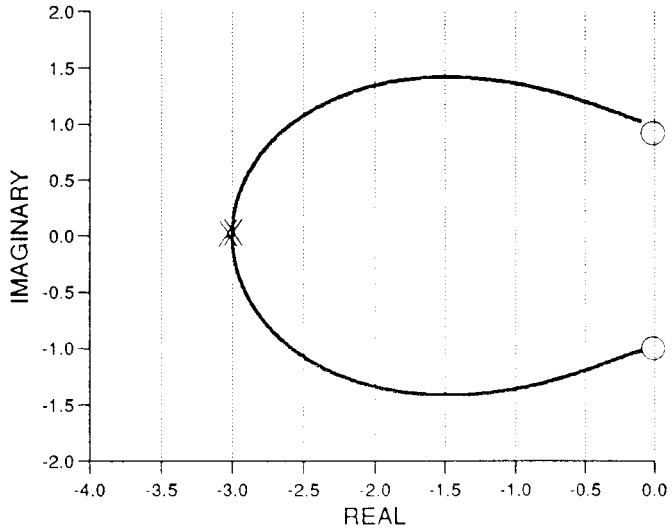
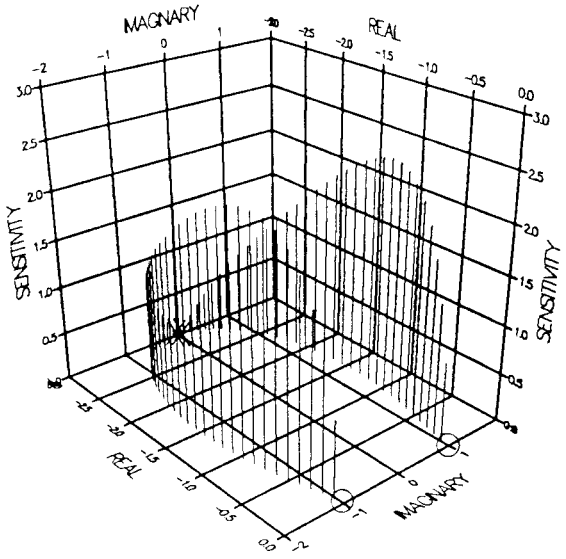


FIGURE 4.2 #K SECOND-ORDER PLANT

ROOT LOCUS

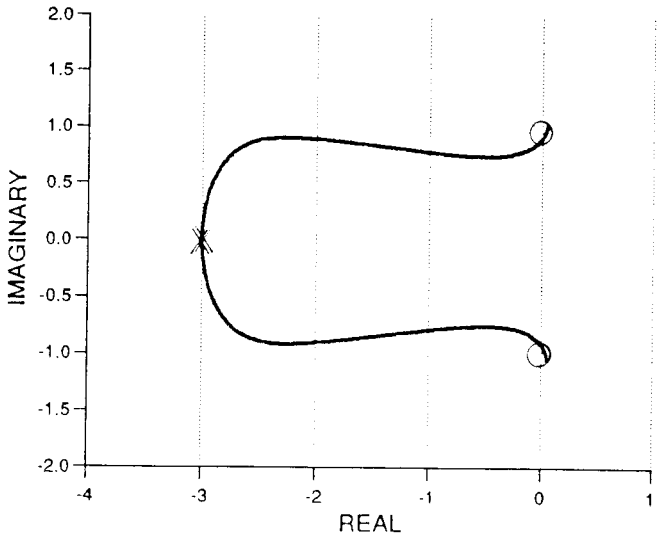


3-D SENSITIVITY PROFILE

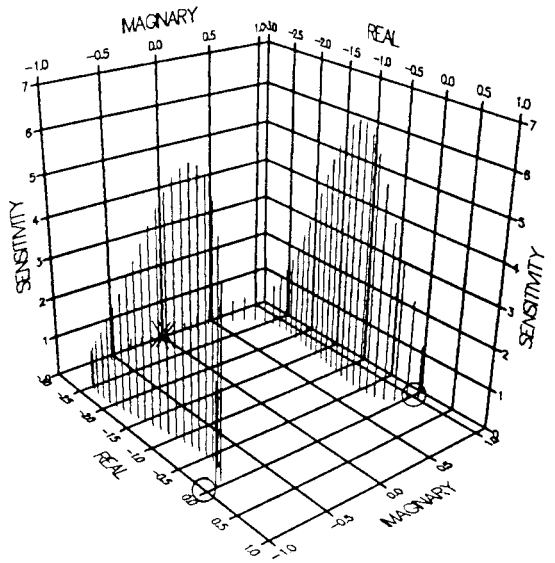


0.2 sec Delay

ROOT LOCUS



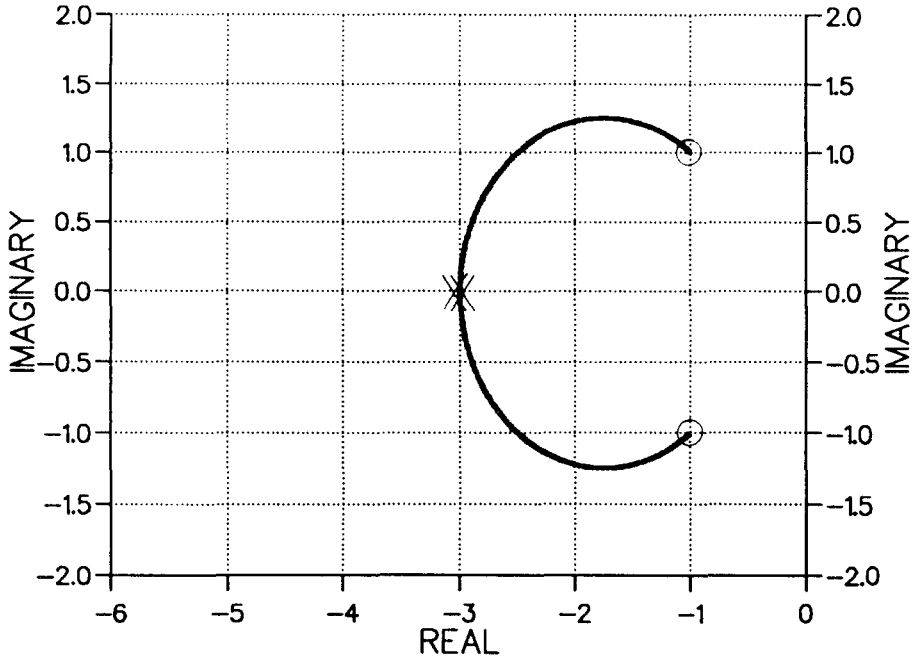
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 #k SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

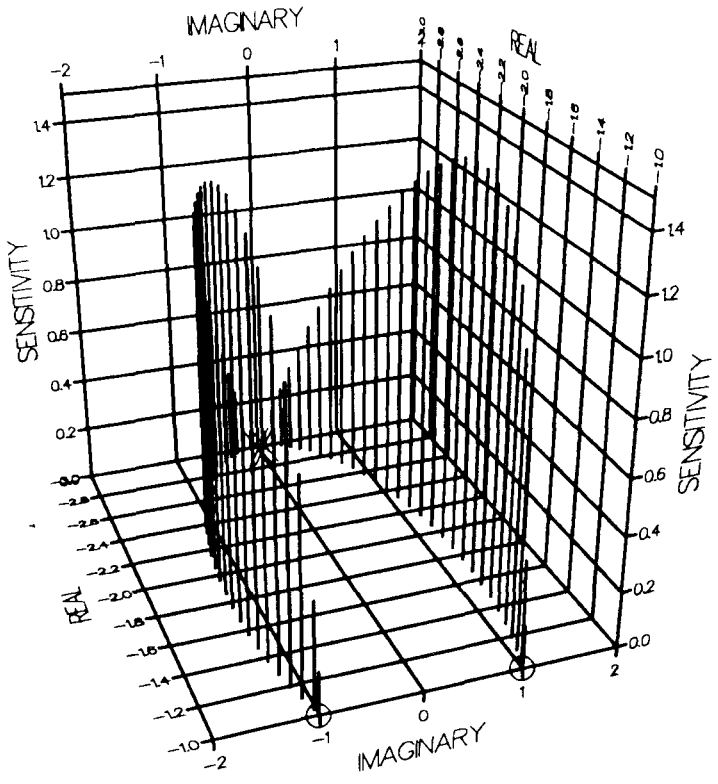
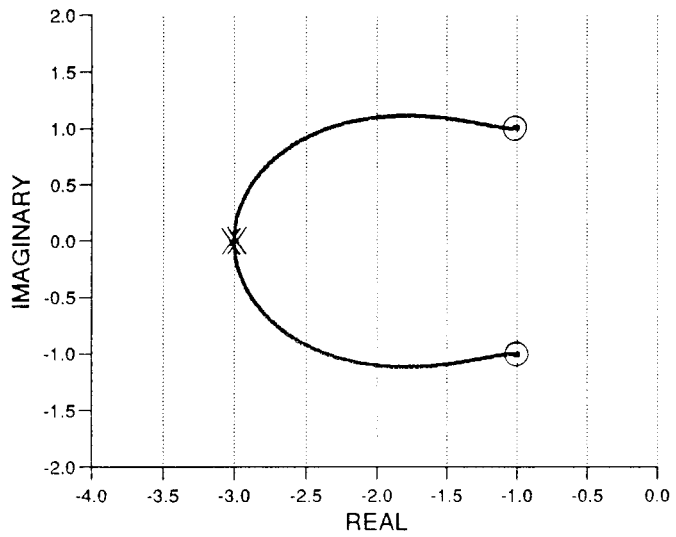
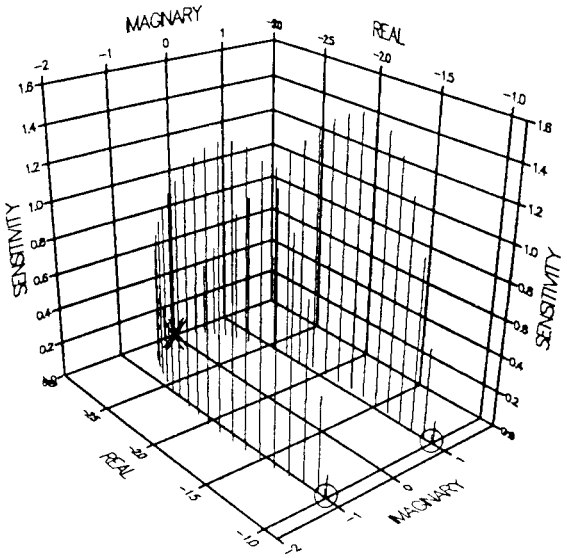


FIGURE 4.2 #L SECOND-ORDER PLANT

ROOT LOCUS

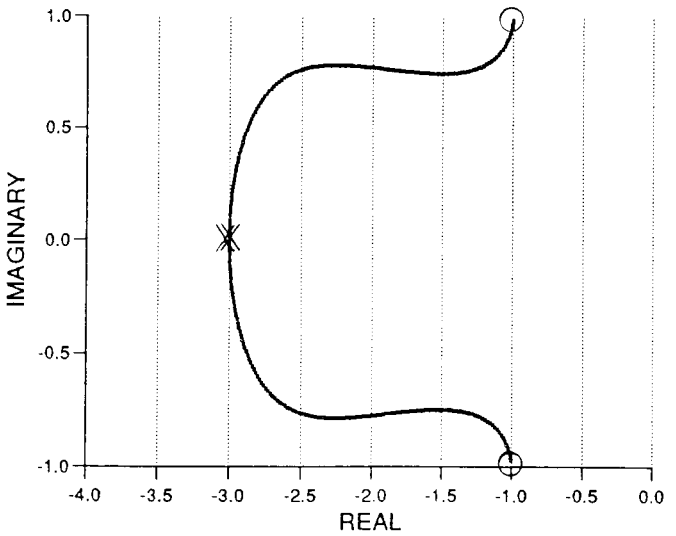


3-D SENSITIVITY PROFILE

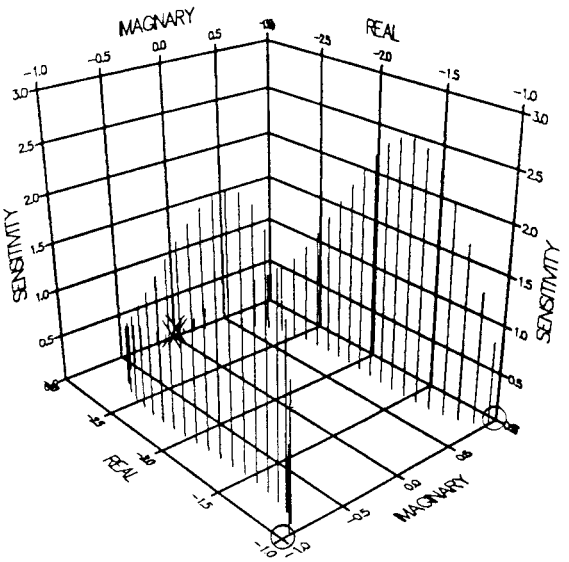


0.2 sec Delay

ROOT LOCUS



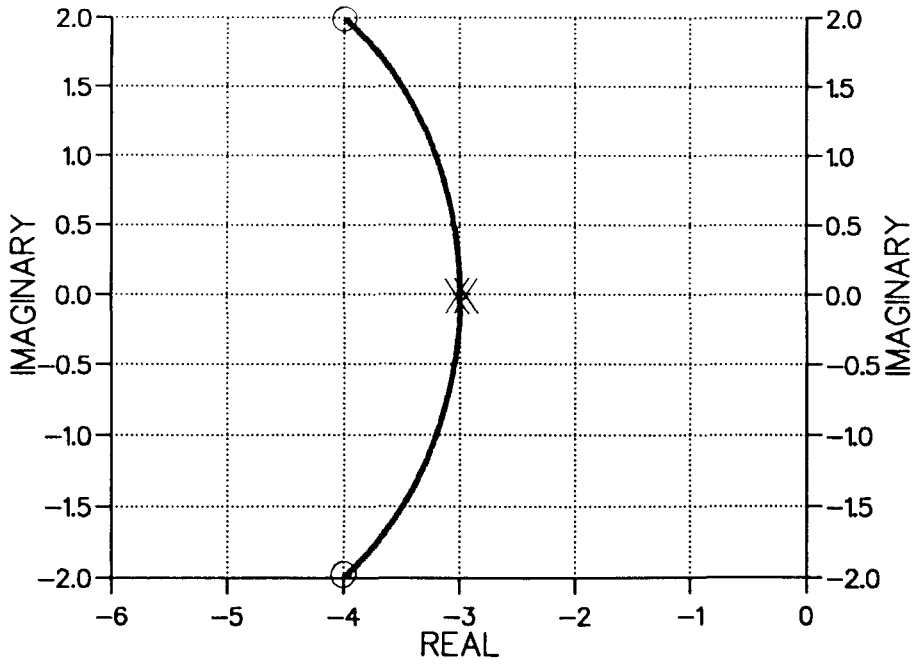
3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 #1 SECOND-ORDER PLANT WITH DELAYS

ROOT LOCUS



3-D SENSITIVITY PROFILE

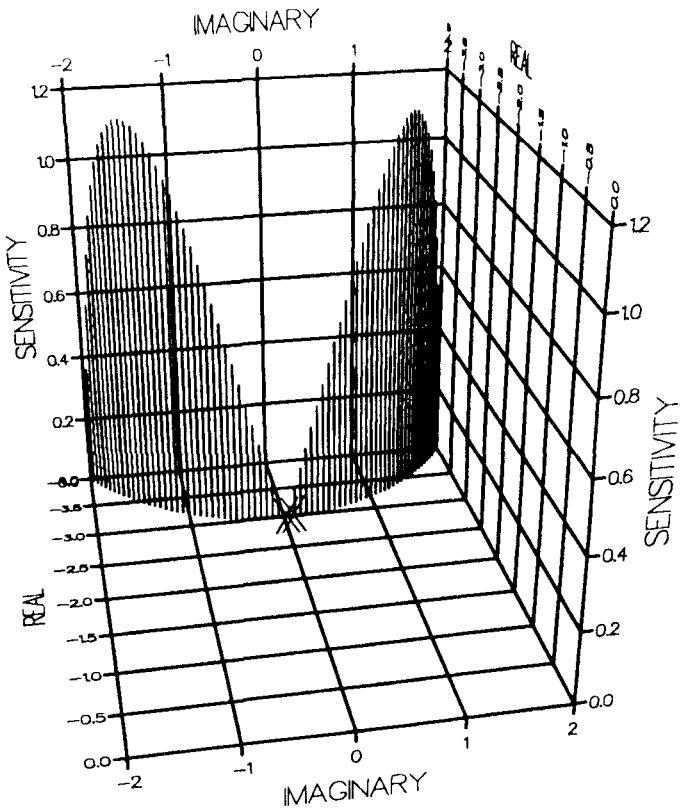
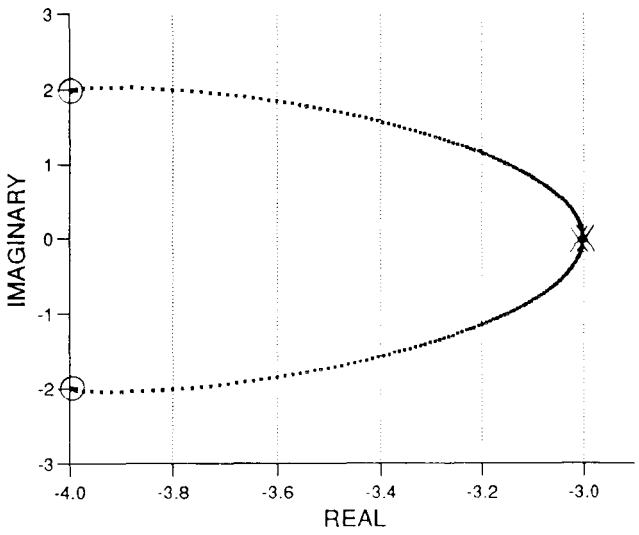
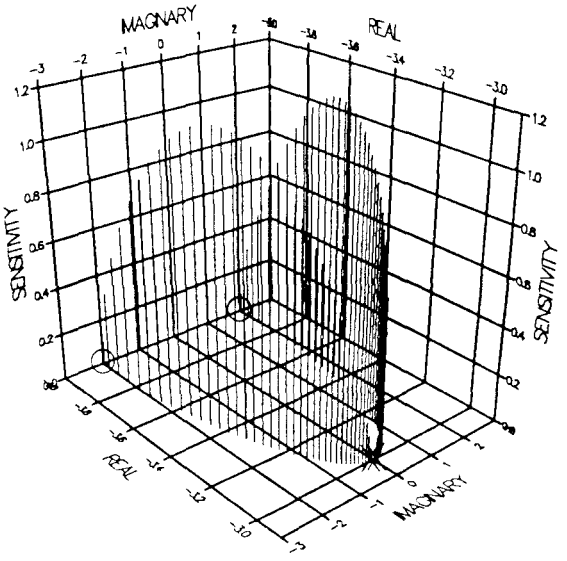


FIGURE 4.2 #M SECOND-ORDER PLANT

ROOT LOCUS

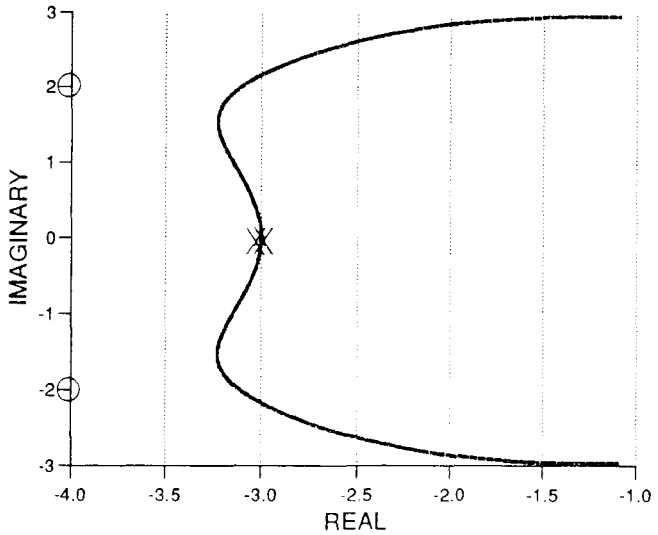


3-D SENSITIVITY PROFILE

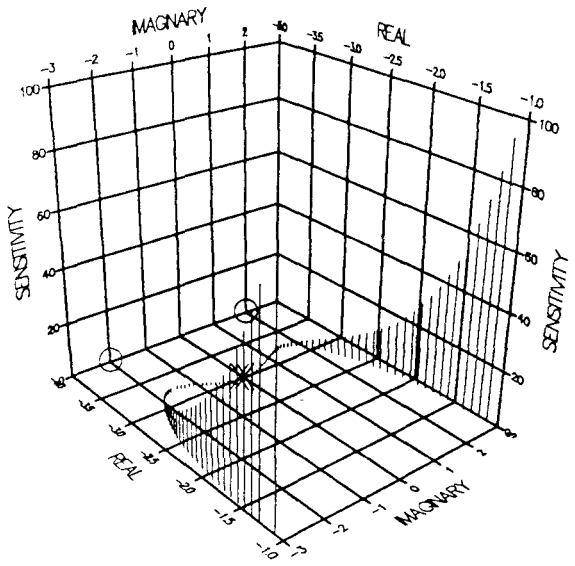


0.2 sec Delay

ROOT LOCUS



3-D SENSITIVITY PROFILE



1.0 sec Delay

FIGURE 4.2 #m SECOND-ORDER PLANT WITH DELAYS

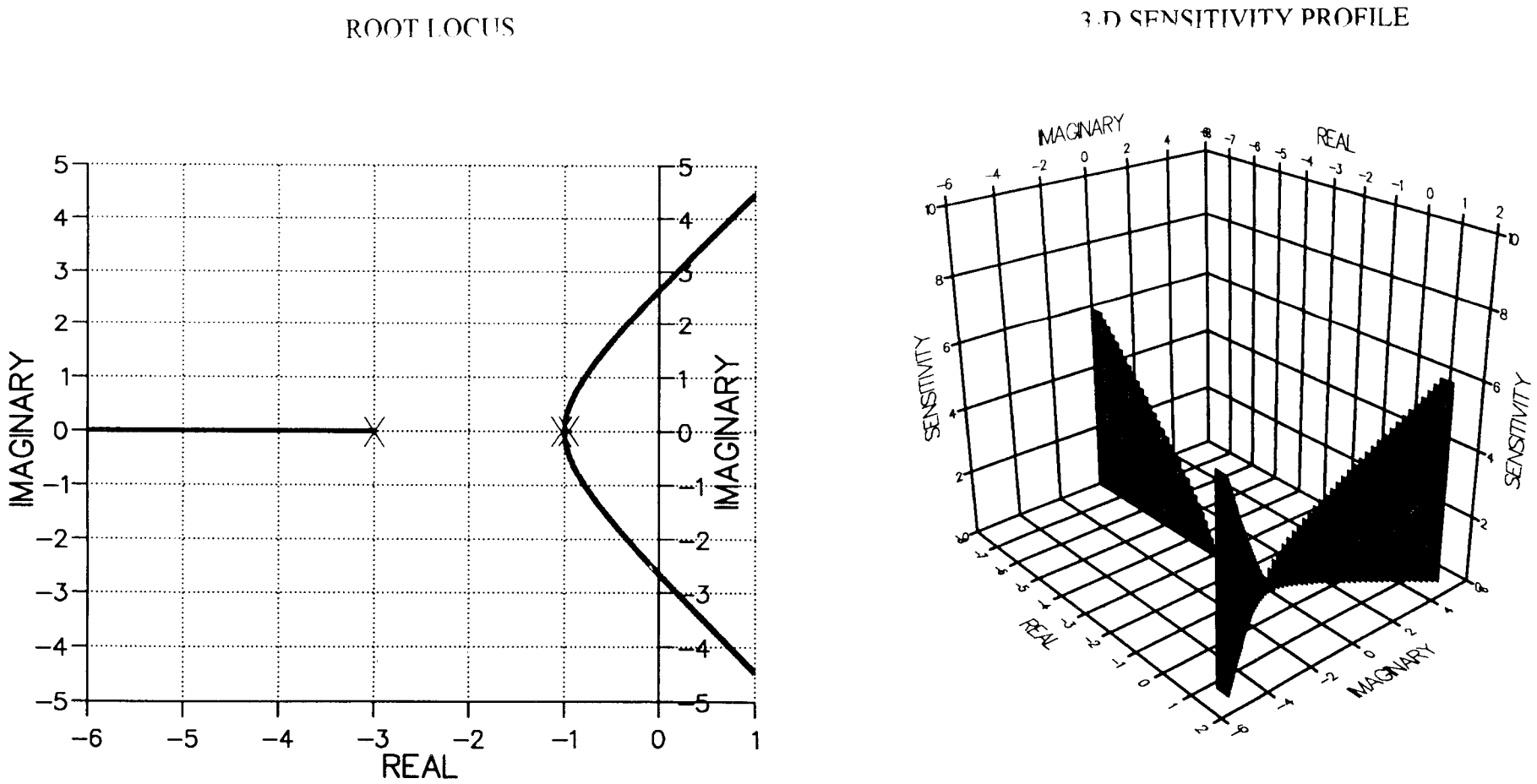


FIGURE 4.3 A THIRD-ORDER PLANT

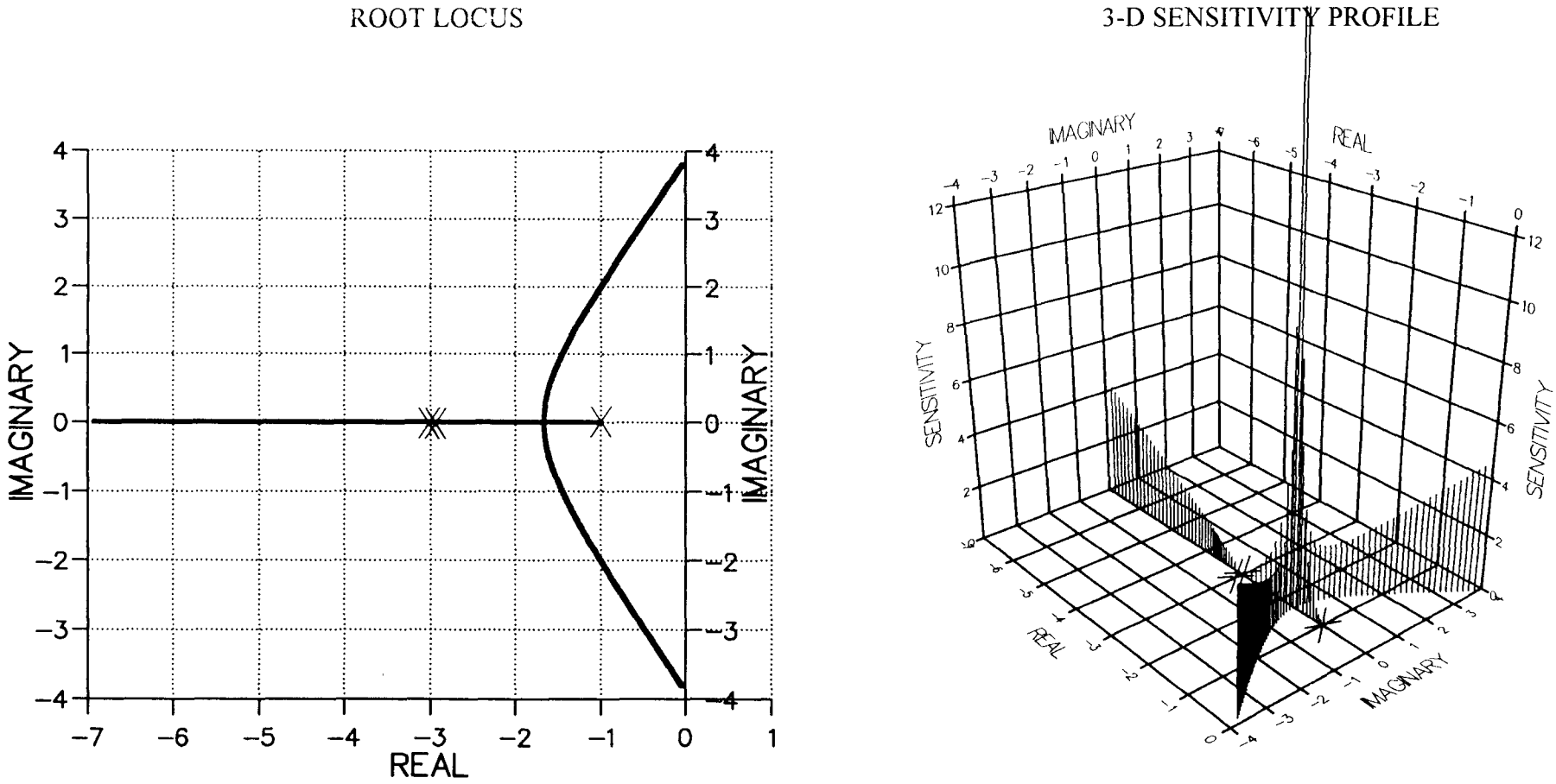


FIGURE 4.3 B THIRD-ORDER PLANT

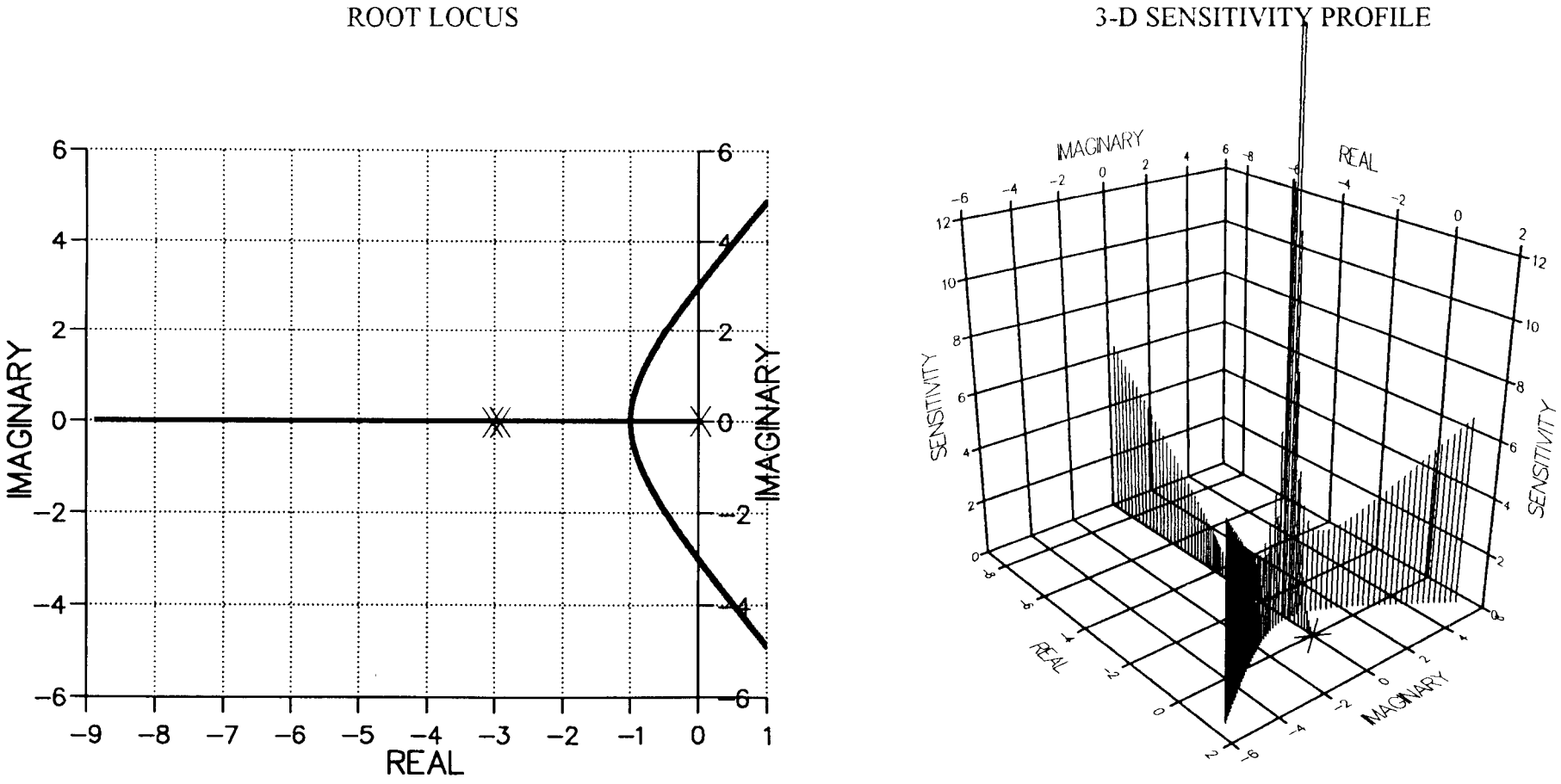
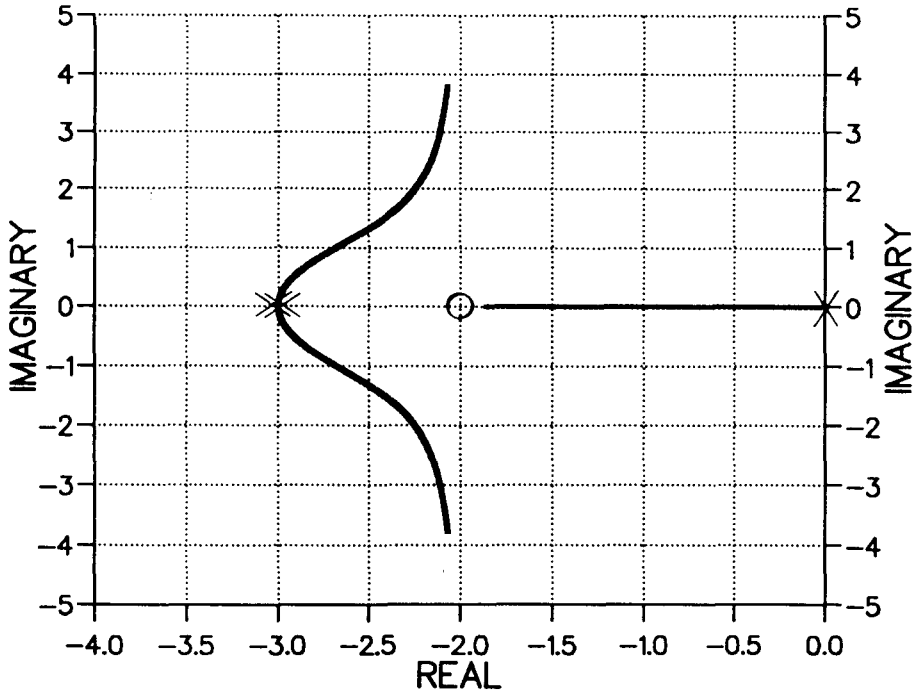


FIGURE 4.3 C THIRD-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

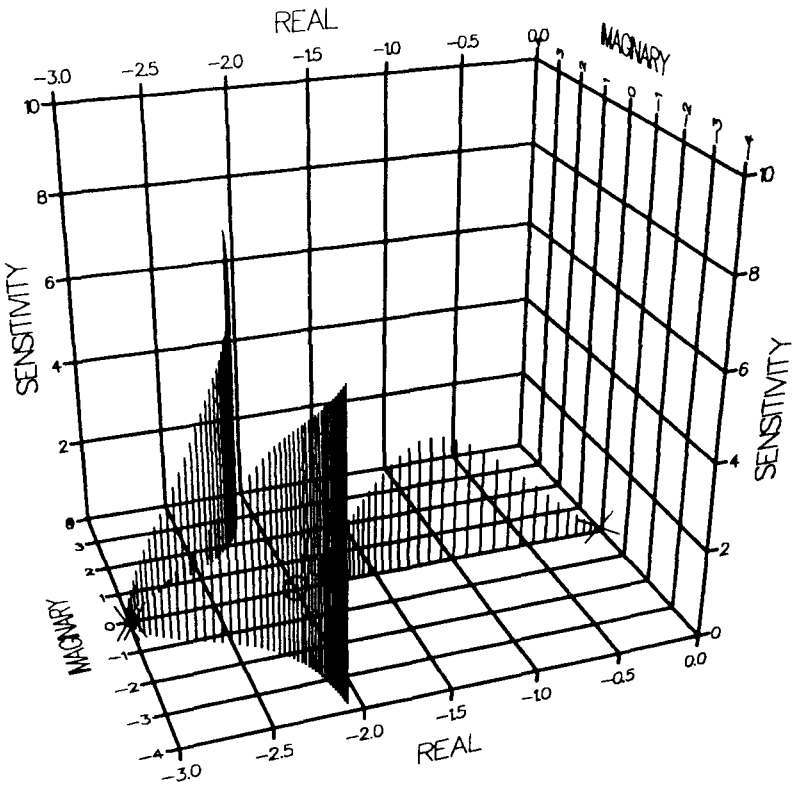
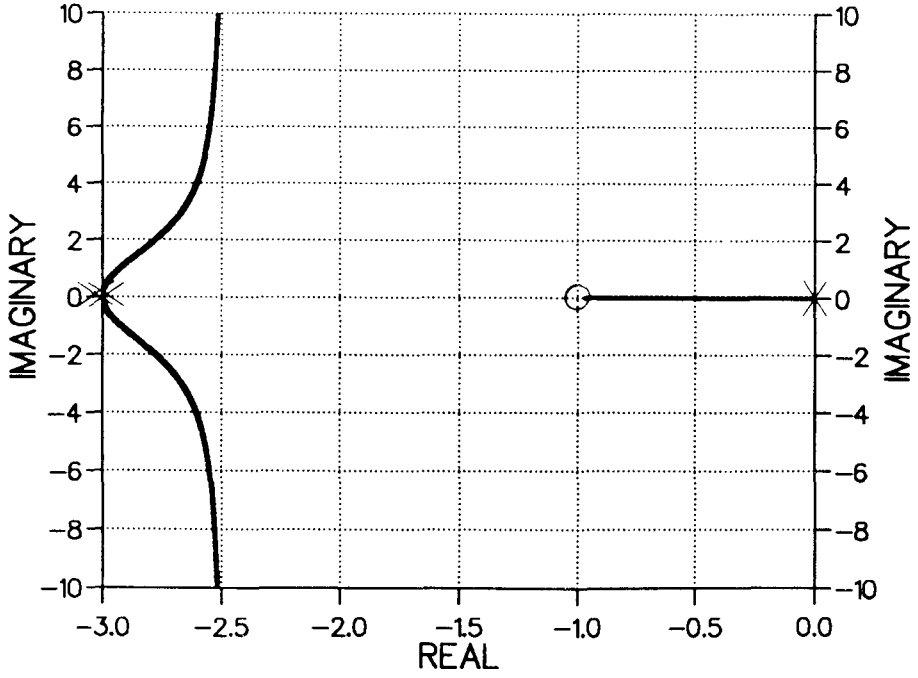


FIGURE 4.3 D THIRD-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

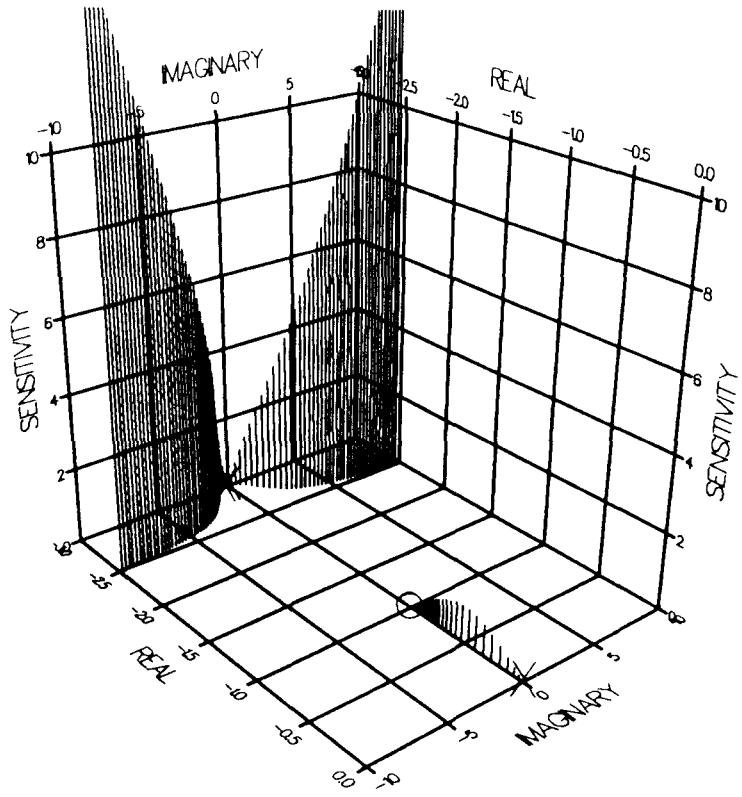
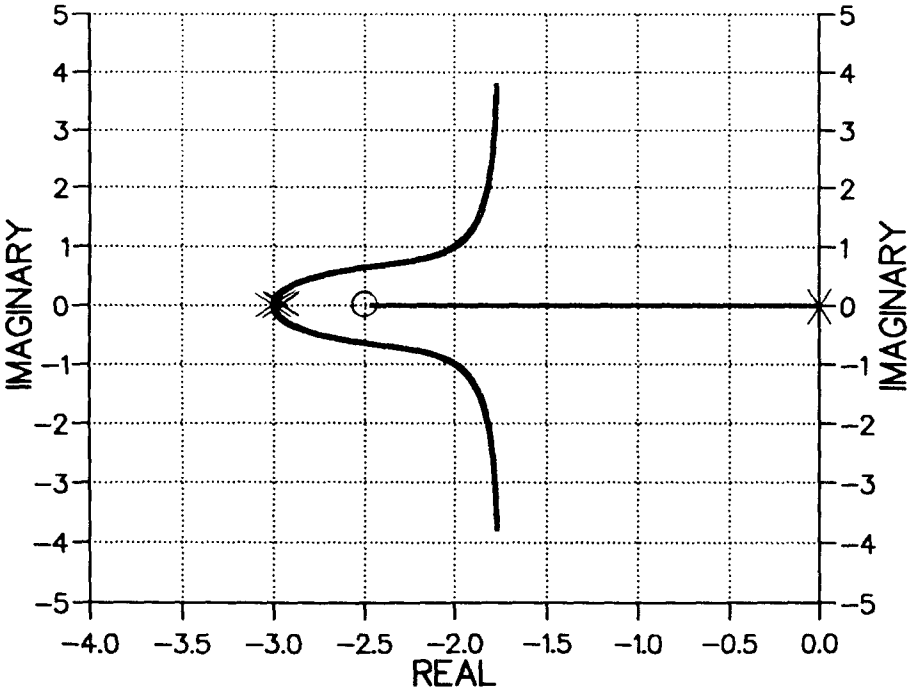
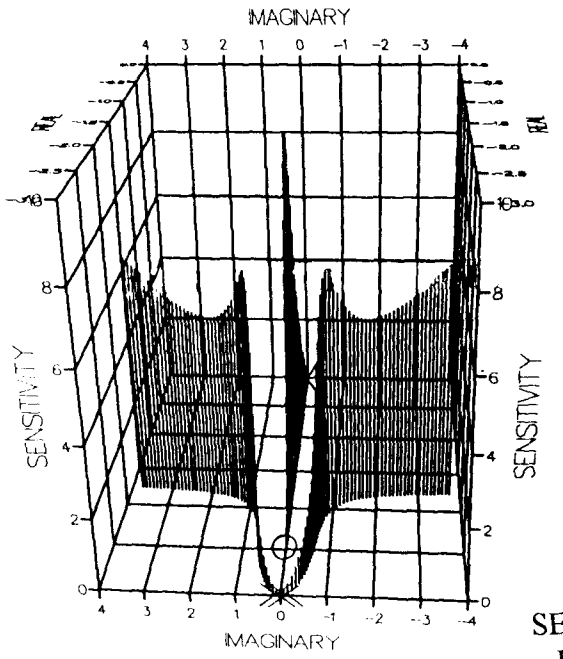


FIGURE 4.3 E THIRD-ORDER PLANT

ROOT LOCUS





3-D SENSITIVITY PROFILES

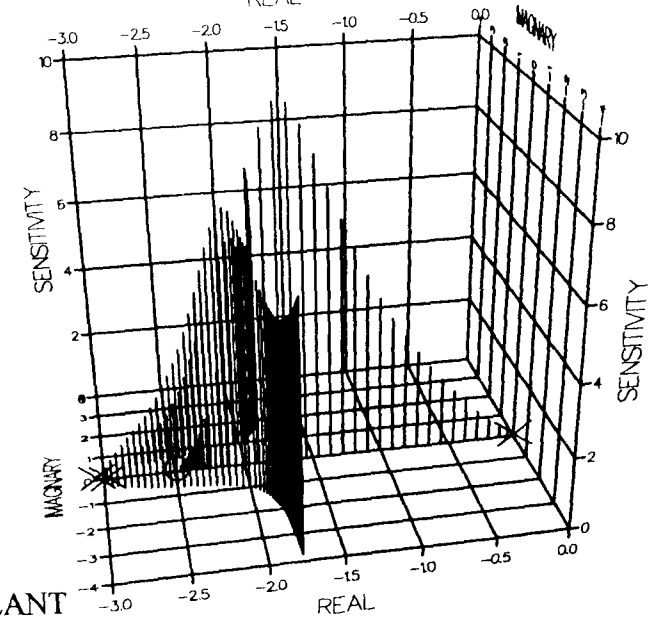


FIGURE 4.3 F THIRD-ORDER PLANT

ROOT LOCUS

3-D SENSITIVITY PROFILE

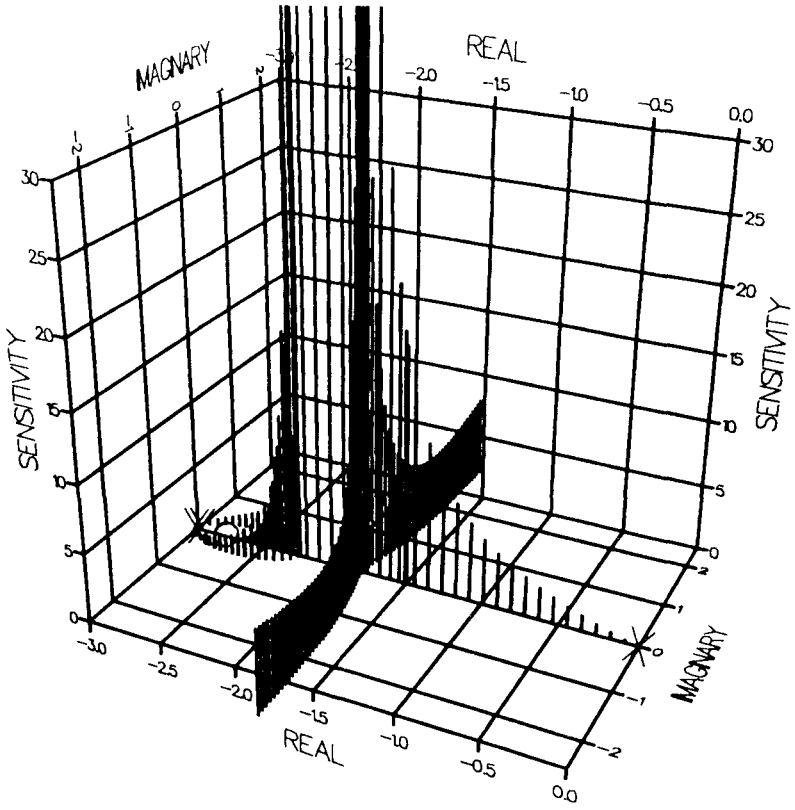
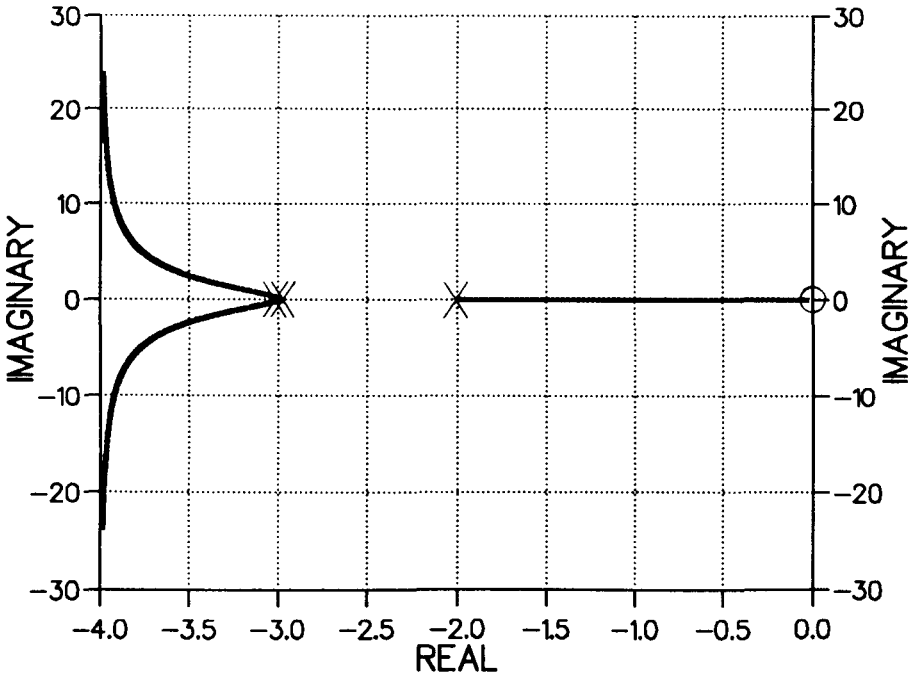


FIGURE 4.3 G THIRD-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

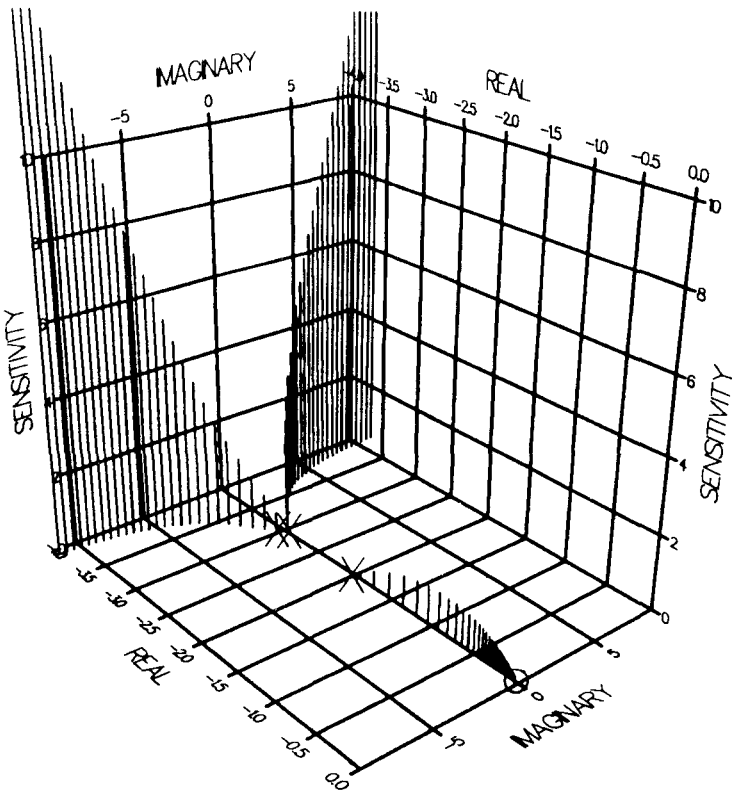
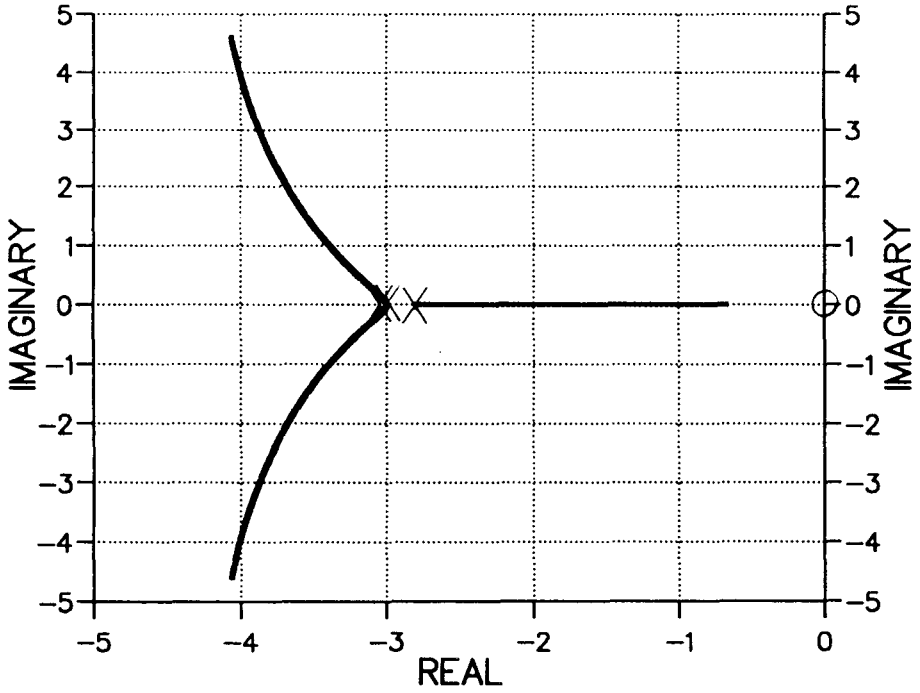


FIGURE 4.3 H THIRD-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

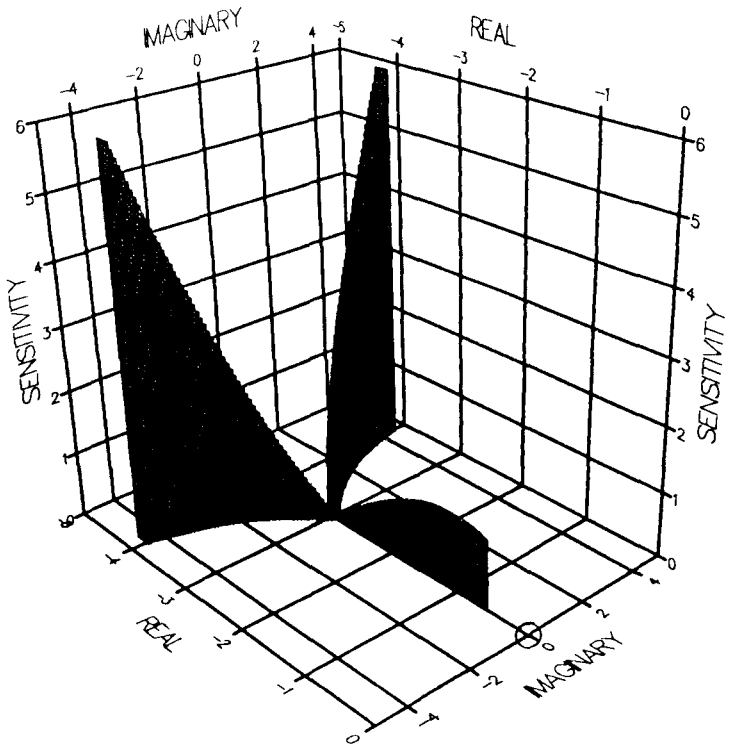


FIGURE 4.3 I THIRD-ORDER PLANT

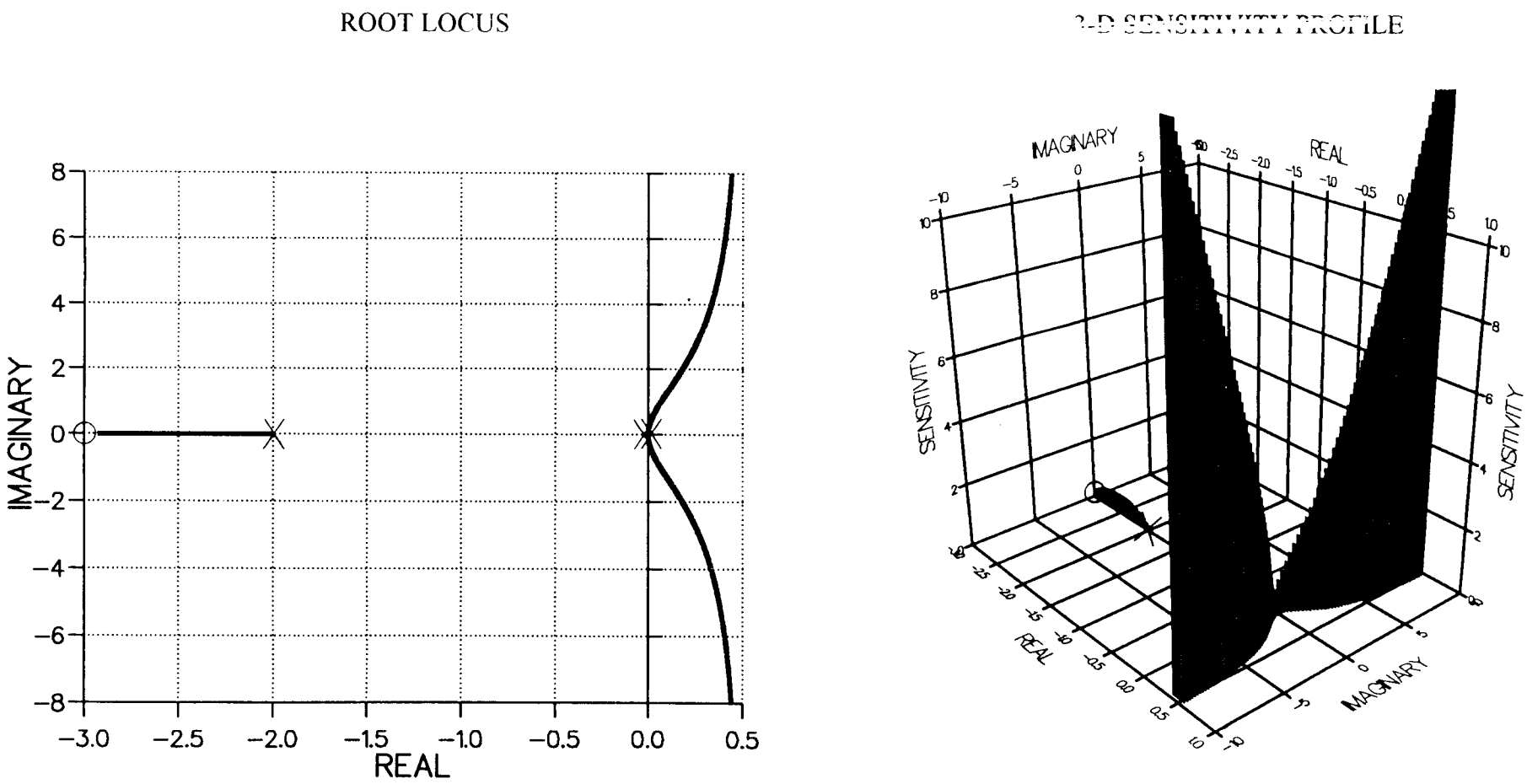


FIGURE 4.3 J THIRD-ORDER PLANT

ROOT LOCUS

3-D SENSITIVITY PROFILE

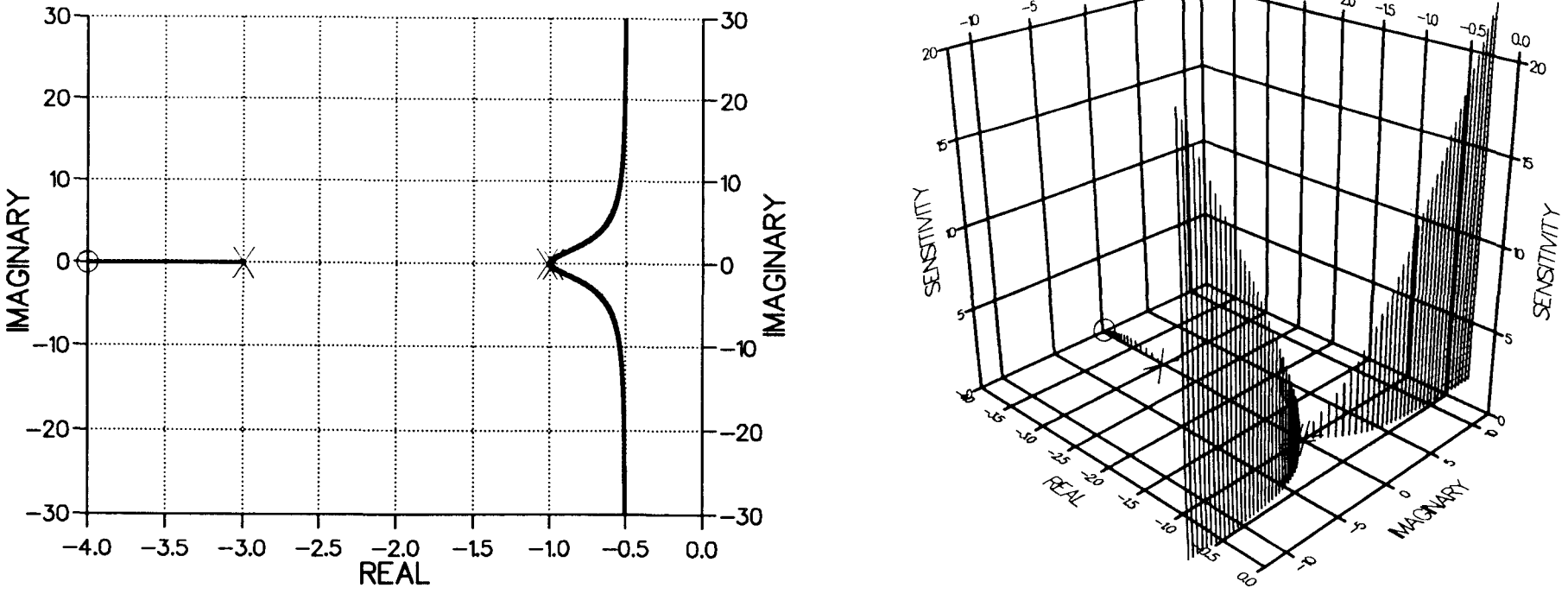
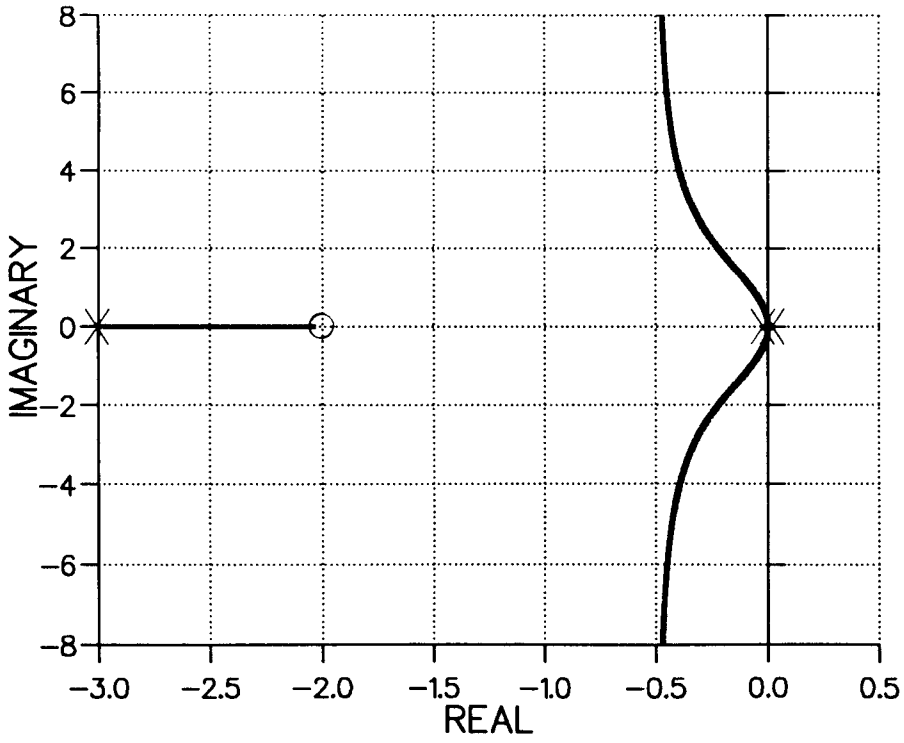


FIGURE 4.3 K THIRD-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

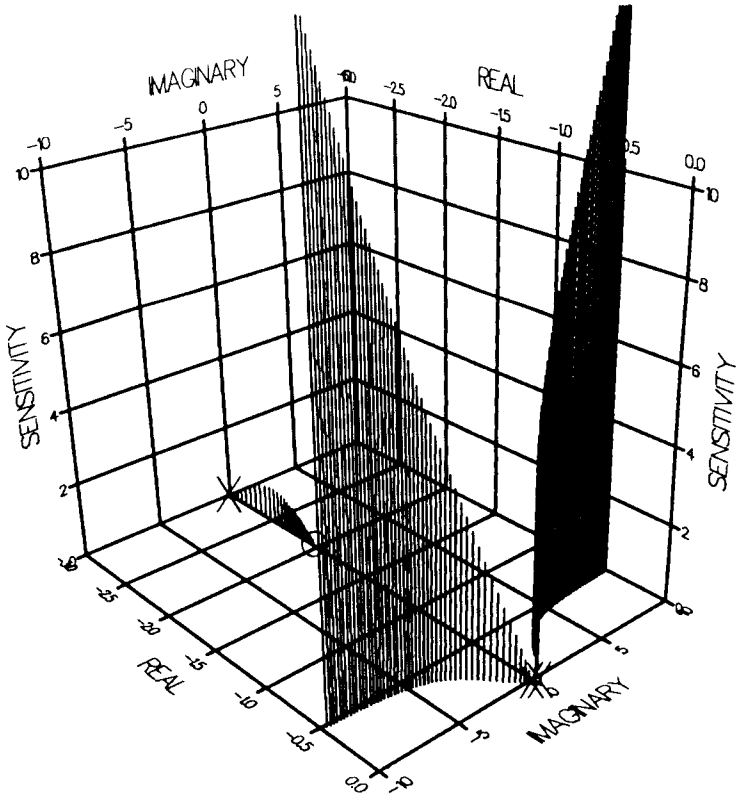
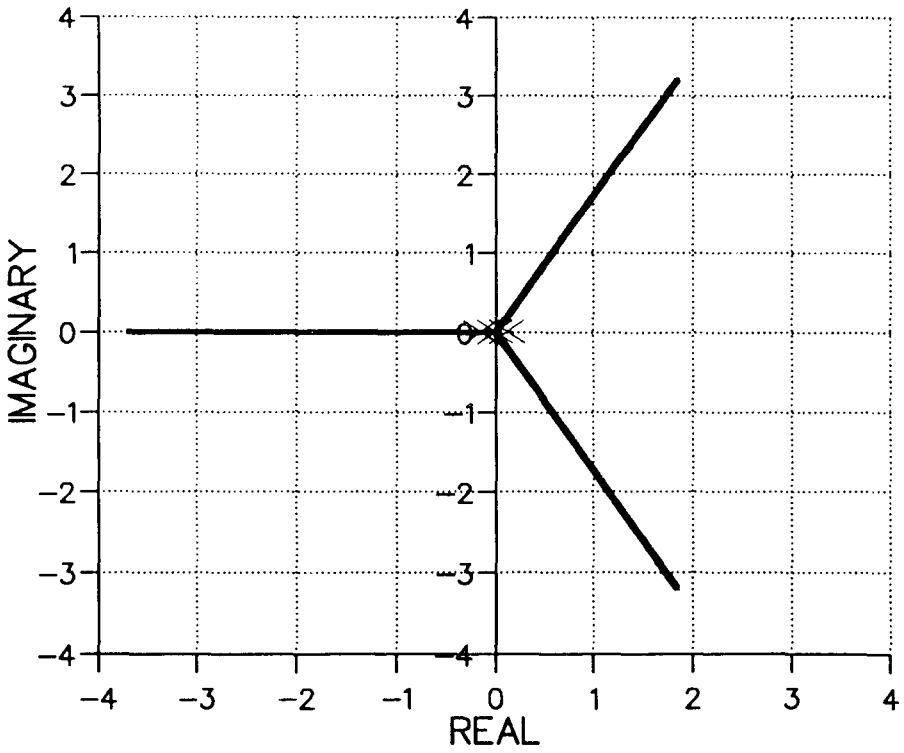


FIGURE 4.3 L THIRD-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

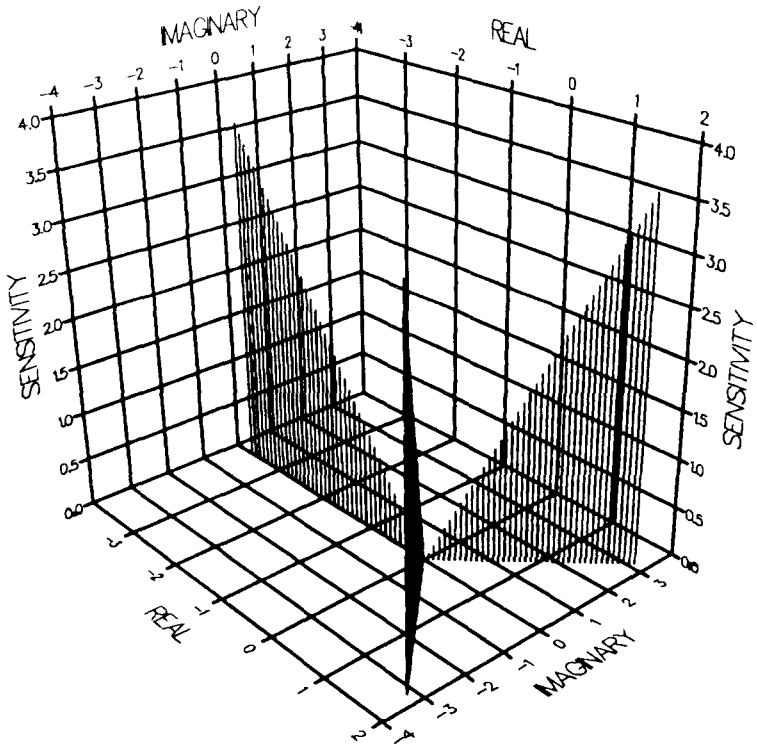
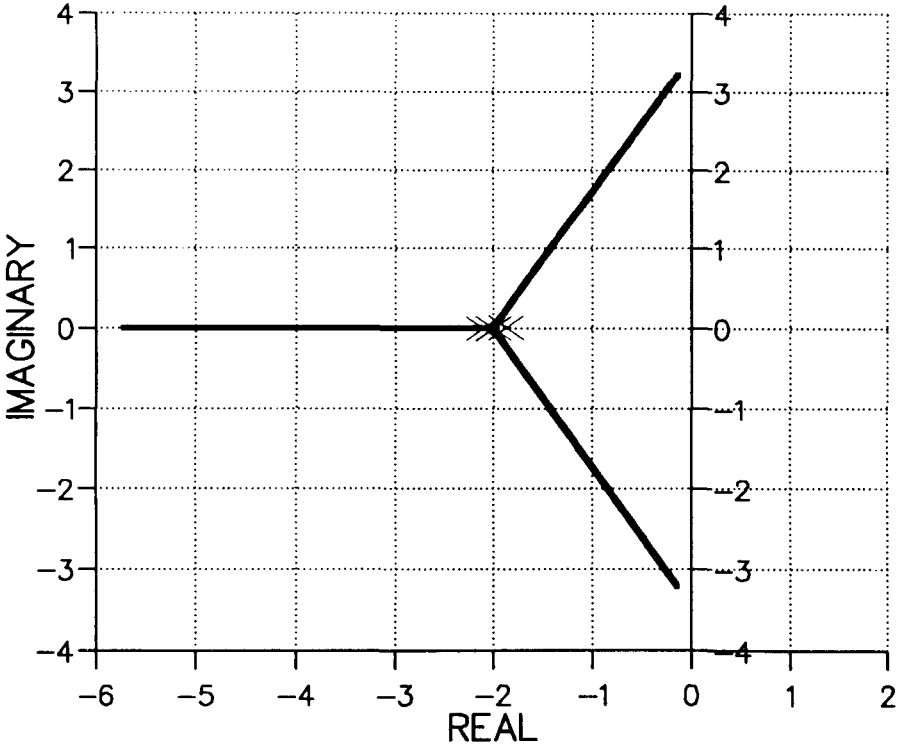


FIGURE 4.3 M THIRD-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

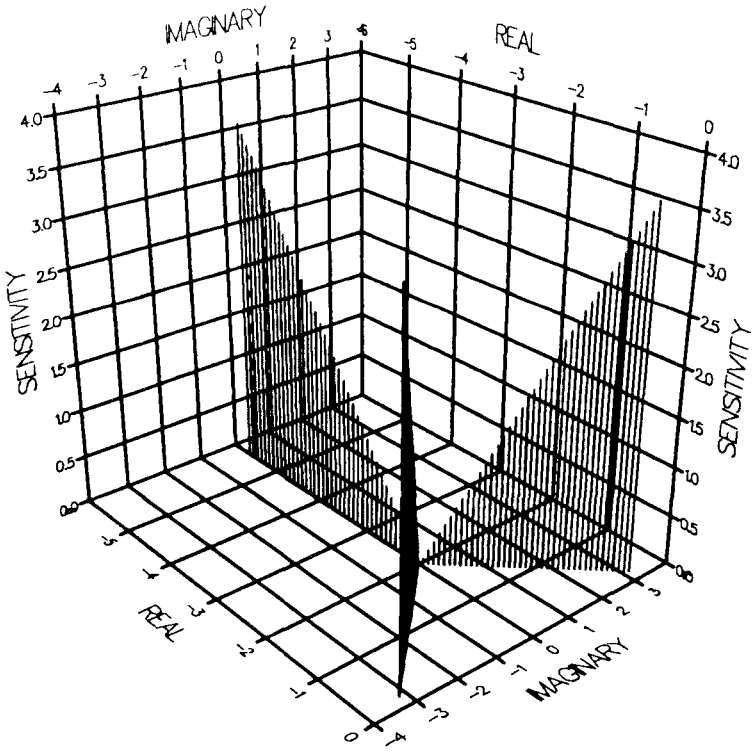


FIGURE 4.3 N THIRD-ORDER PLANT

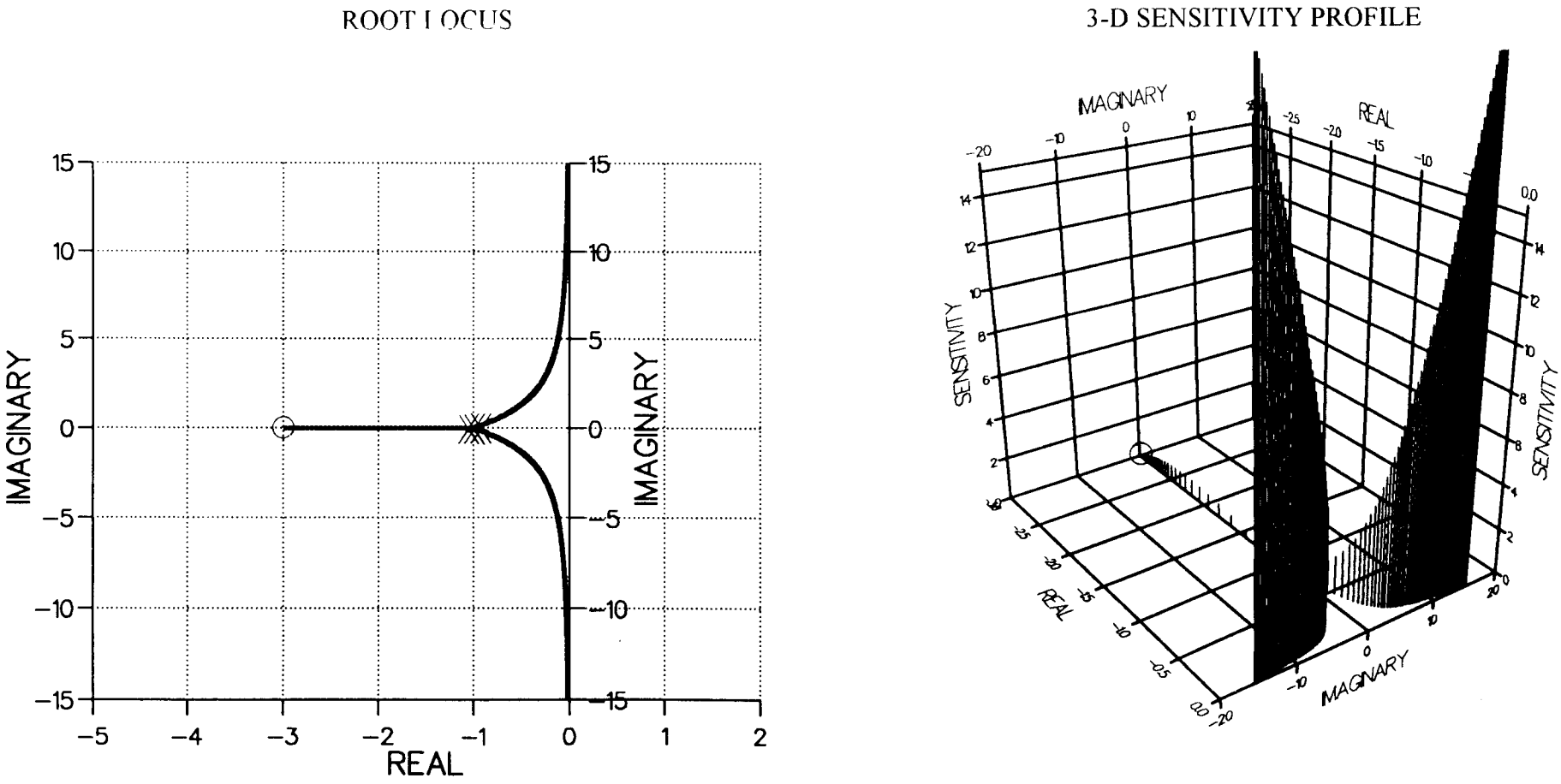
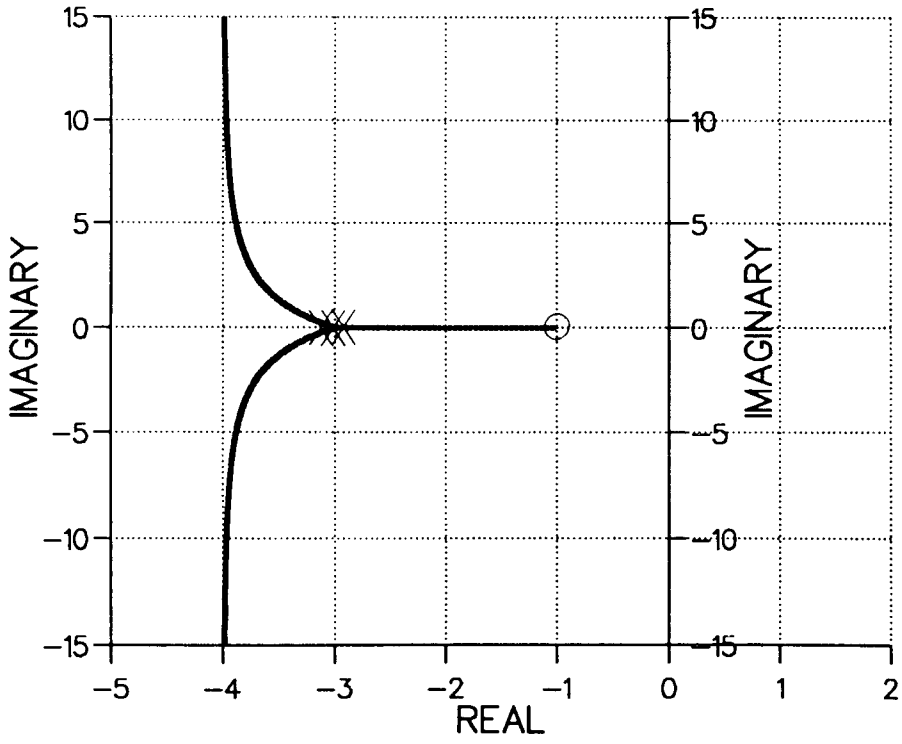


FIGURE 4.3 O THIRD-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

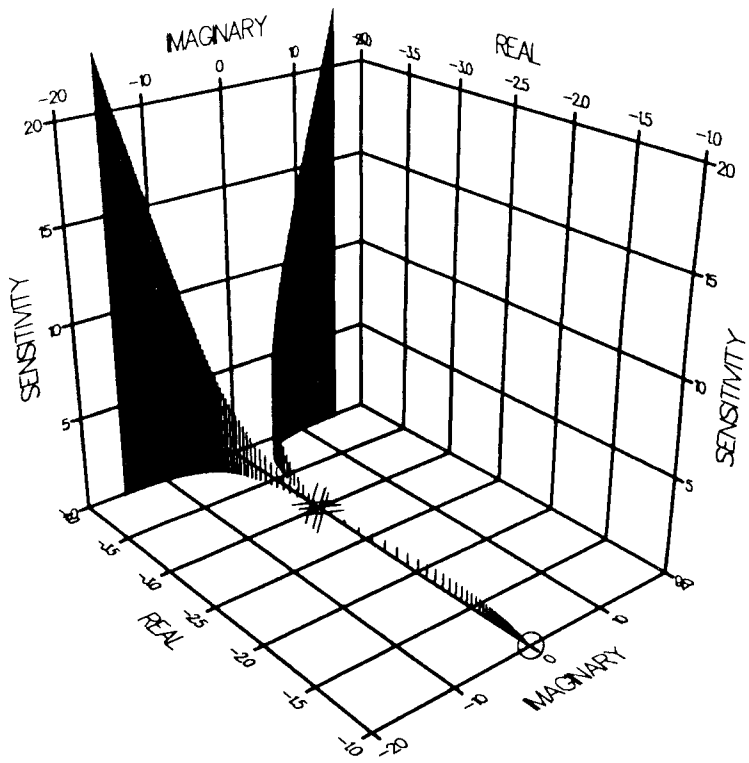
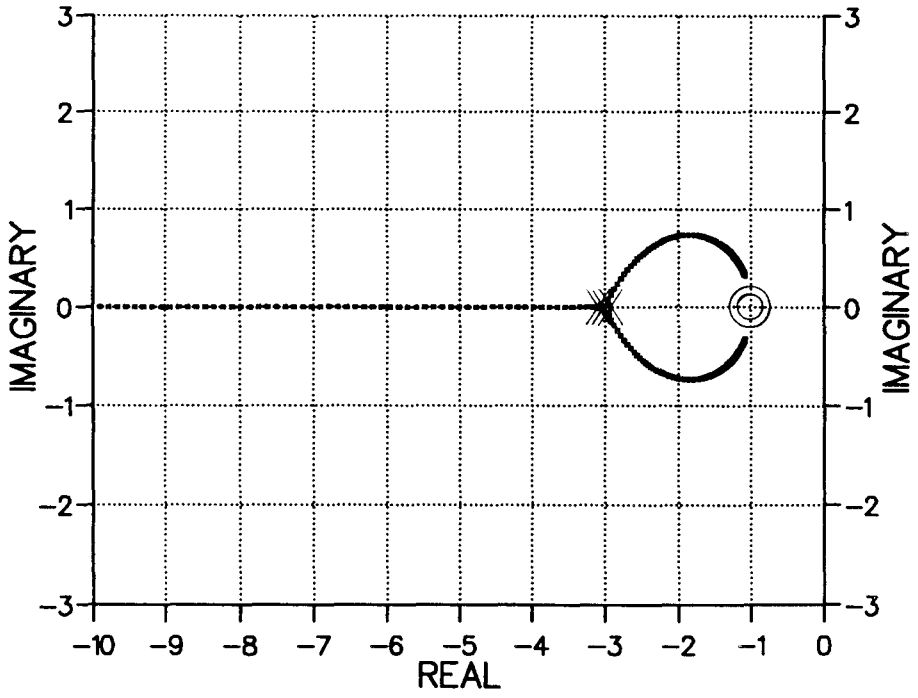


FIGURE 4.3 P THIRD-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

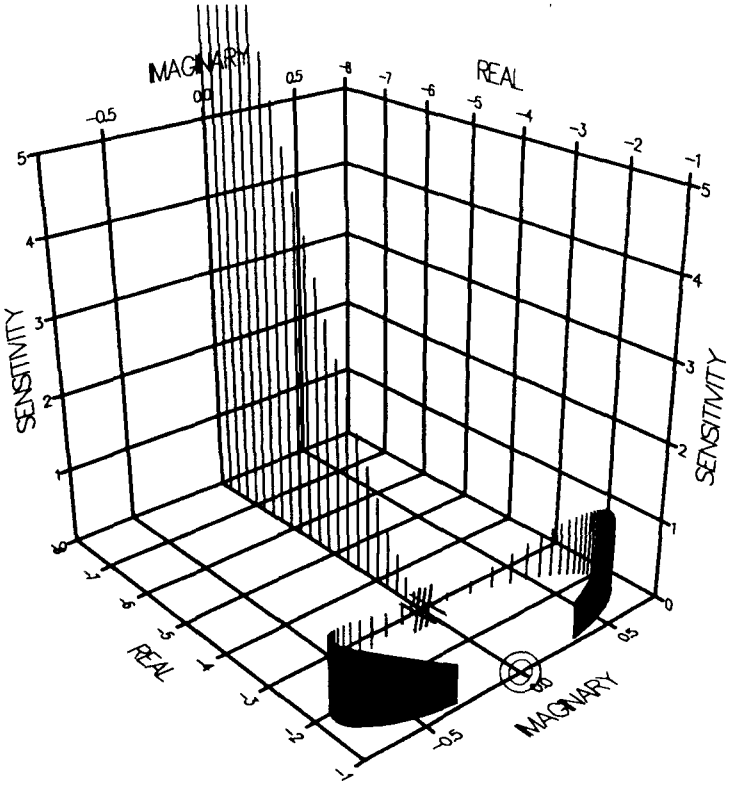
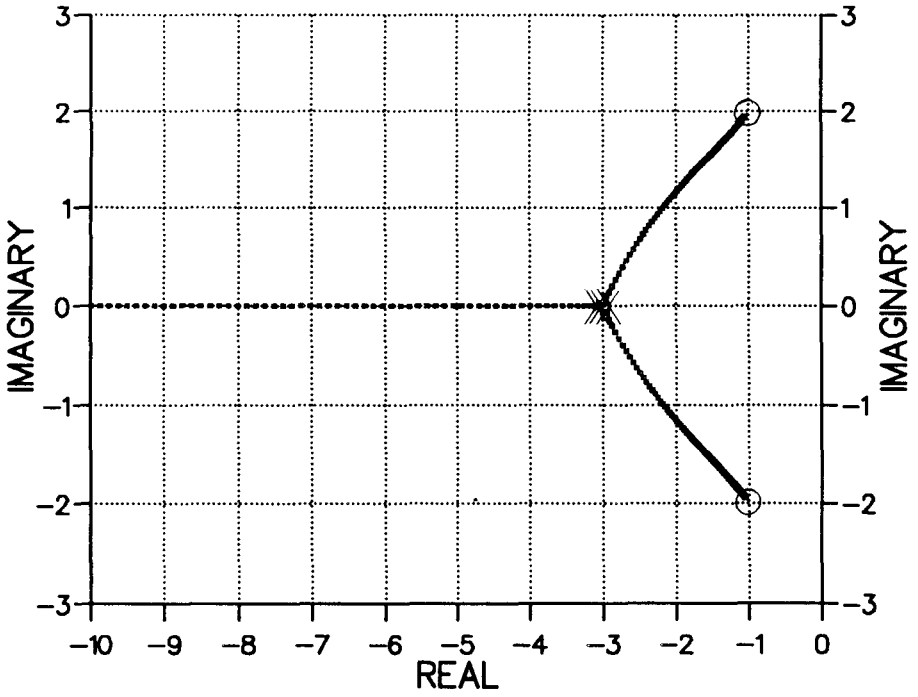


FIGURE 4.3 Q THIRD-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

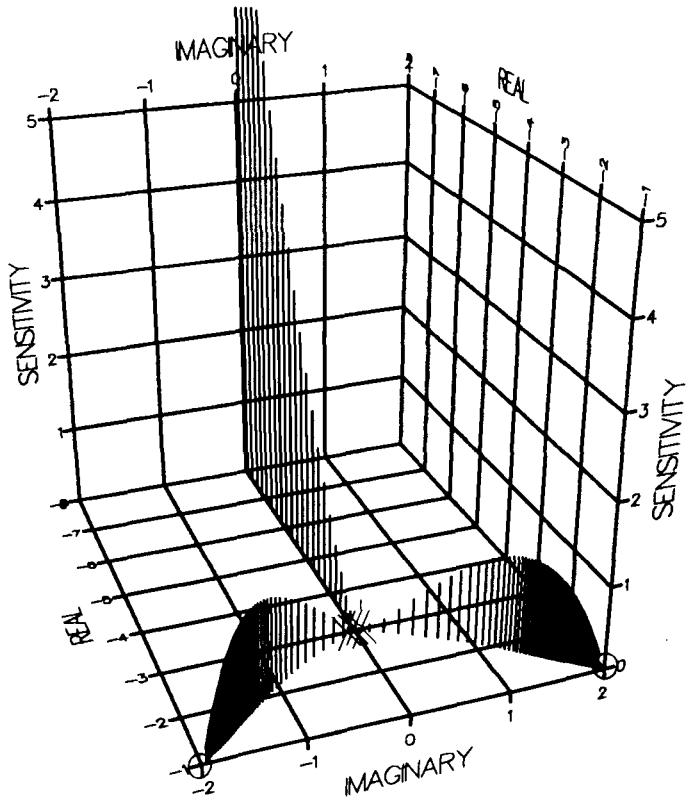
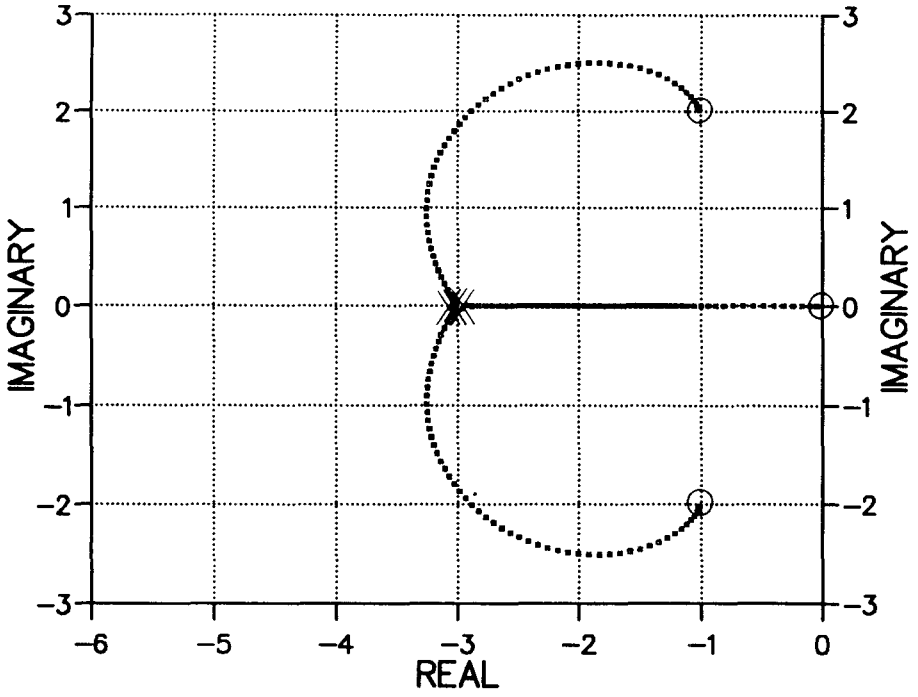


FIGURE 4.3 R THIRD-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

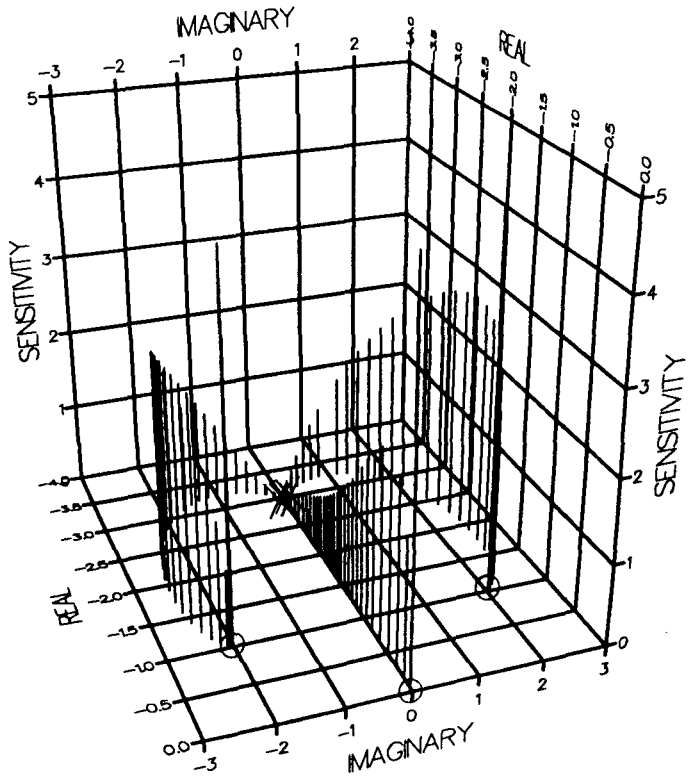
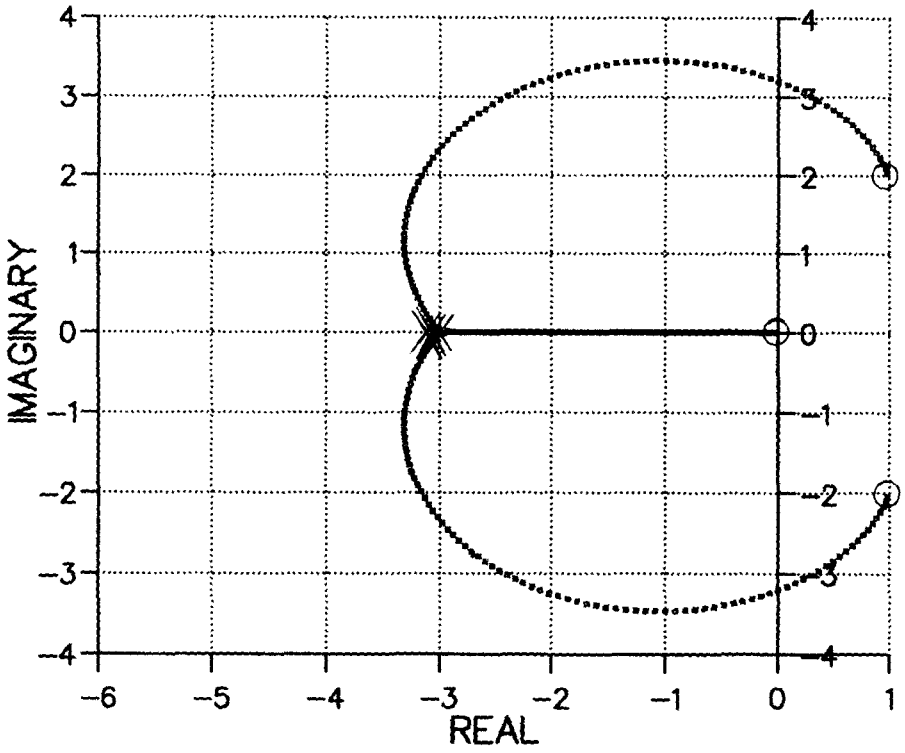


FIGURE 4.3 S THIRD-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

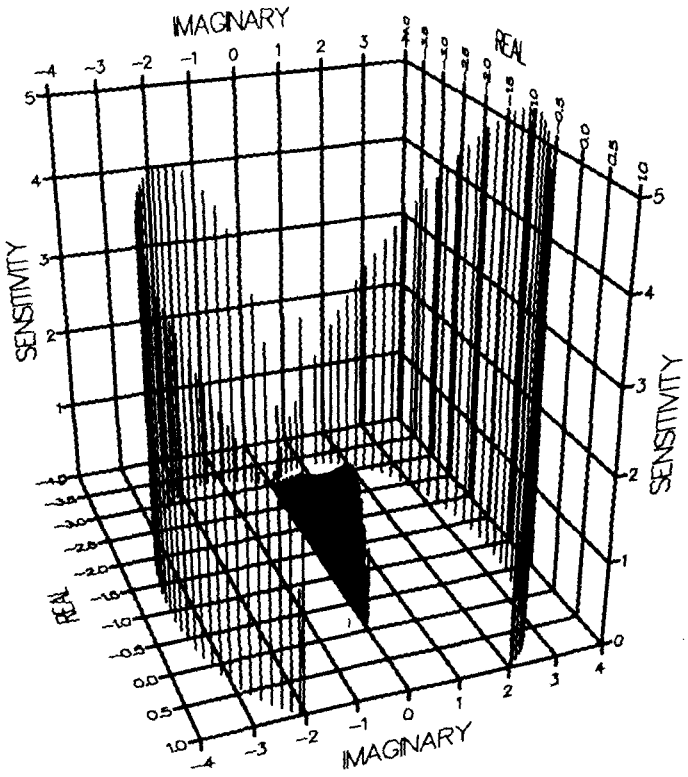
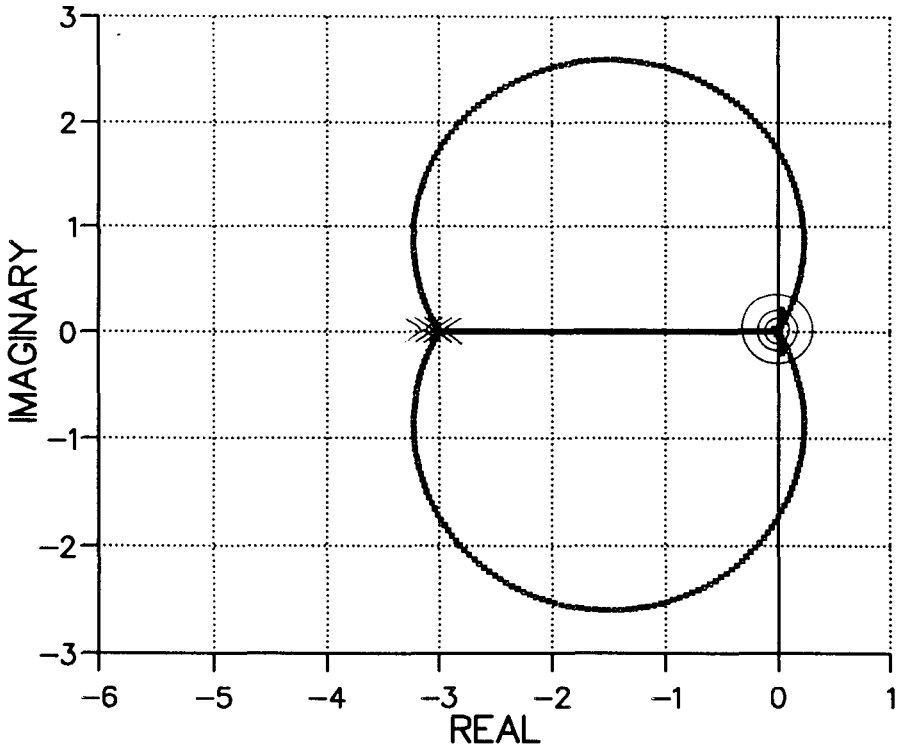


FIGURE 4.3 T THIRD-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

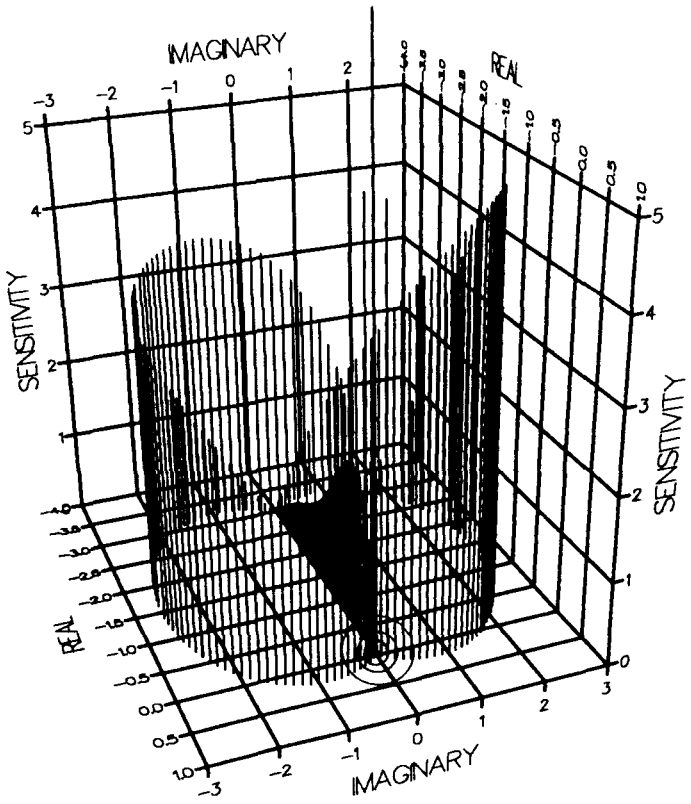
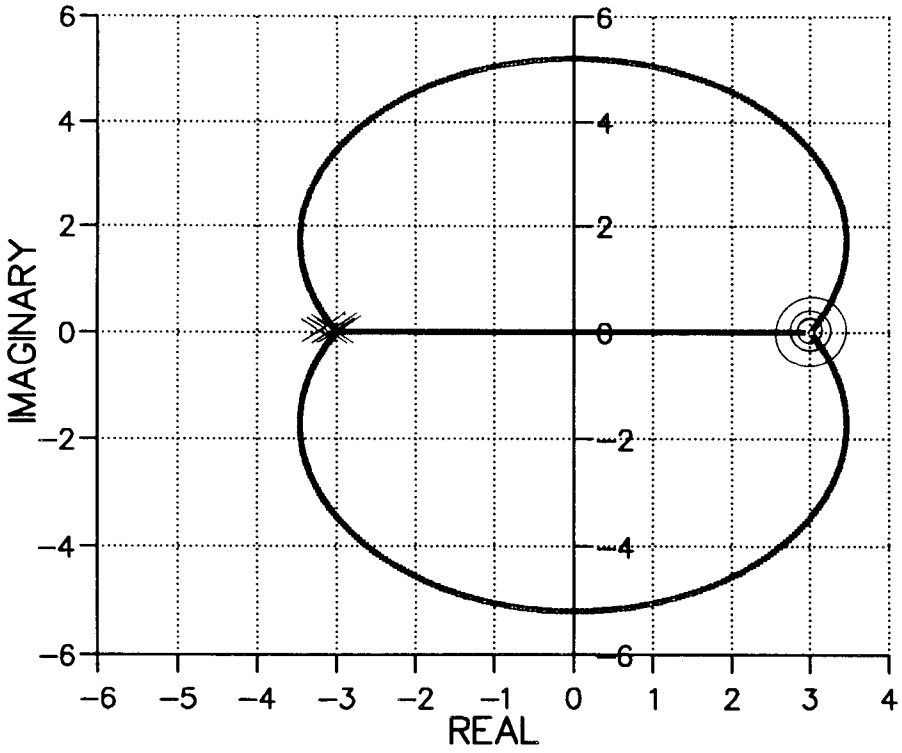


FIGURE 4.3 U THIRD-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

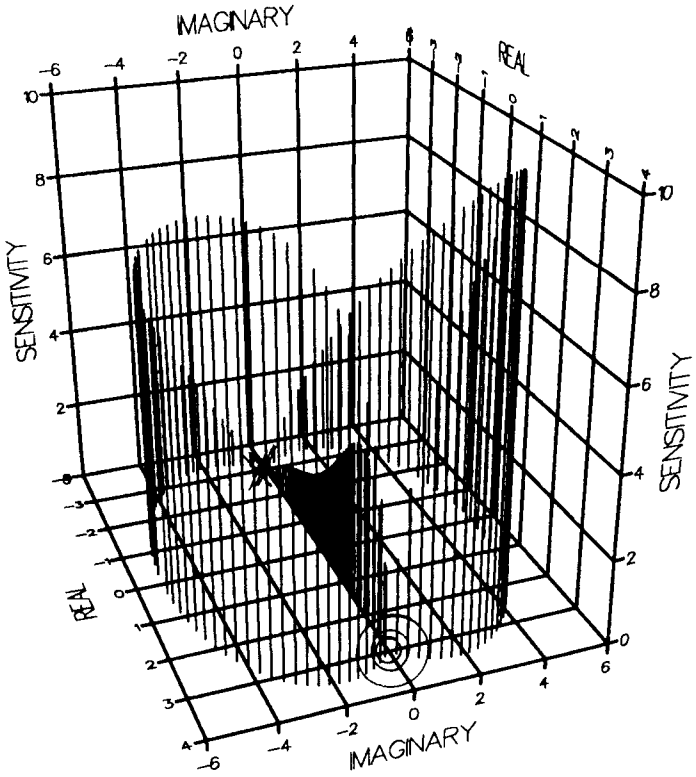


FIGURE 4.3 V THIRD-ORDER PLANT

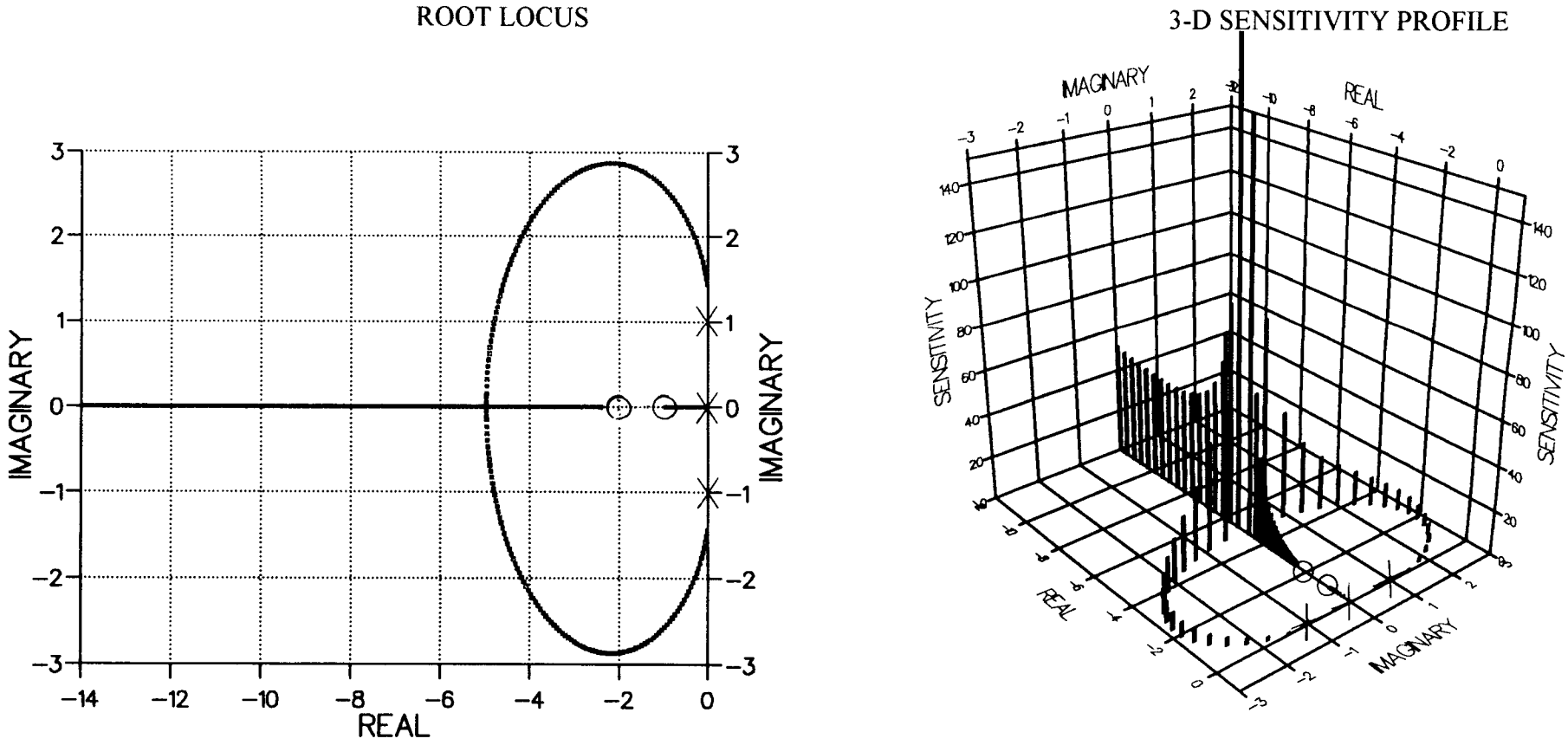
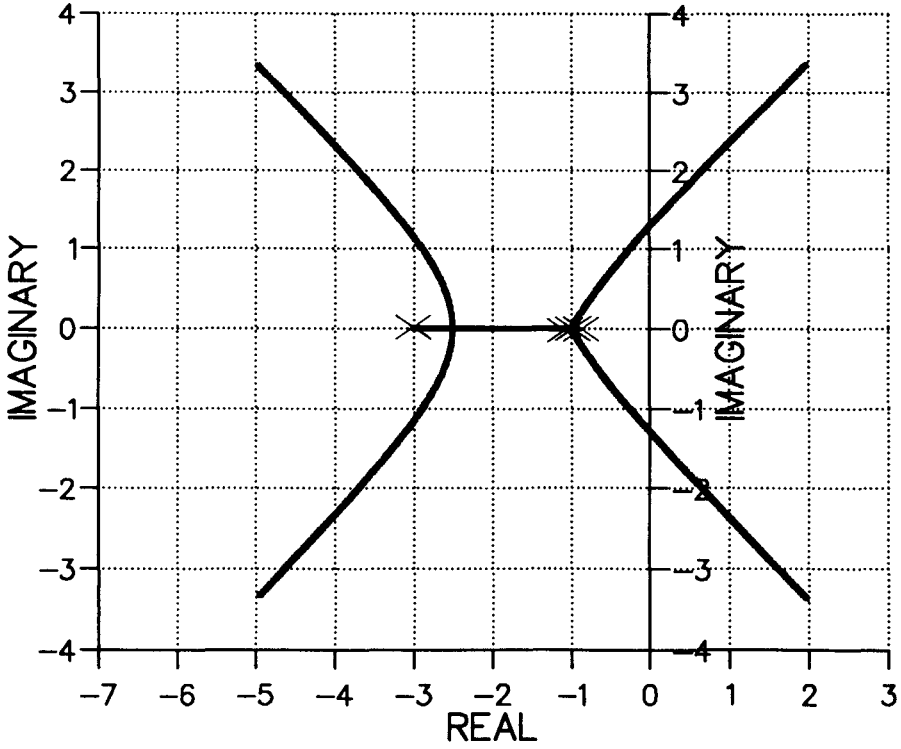


FIGURE 4.3 W THIRD-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

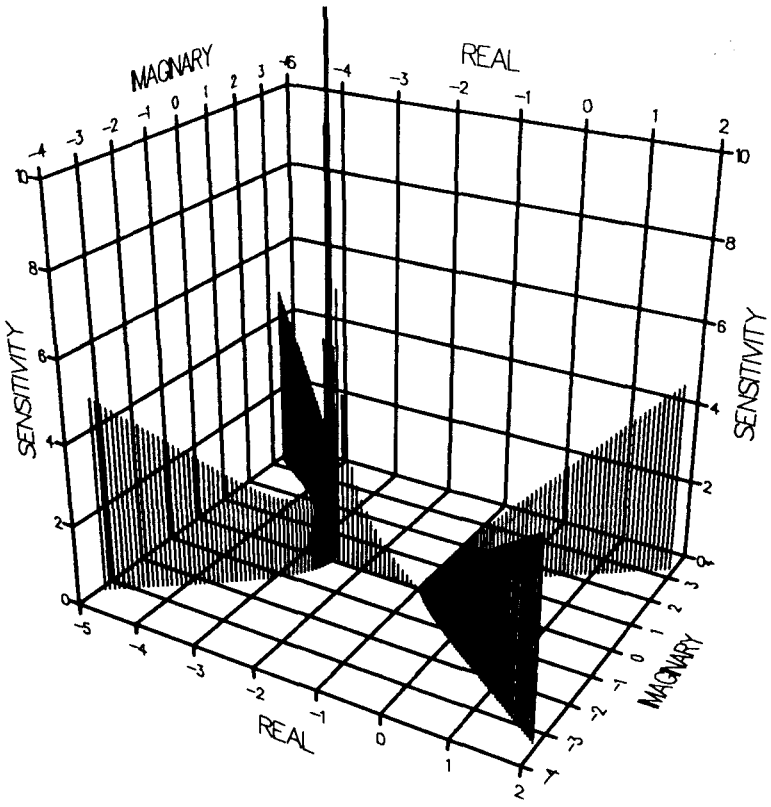
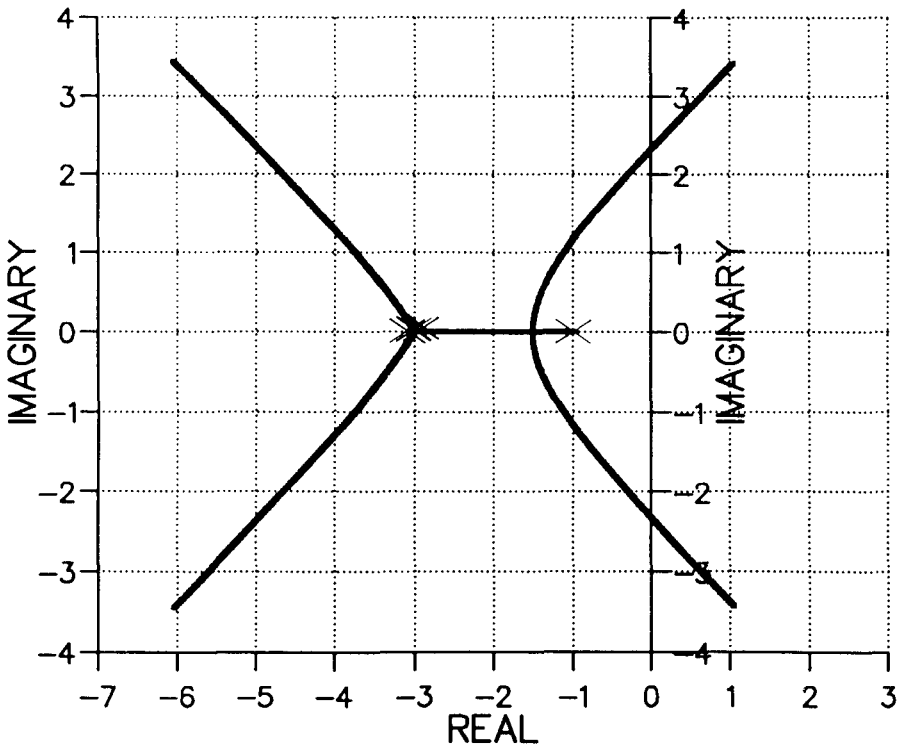


FIGURE 4.4 A FOURTH-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

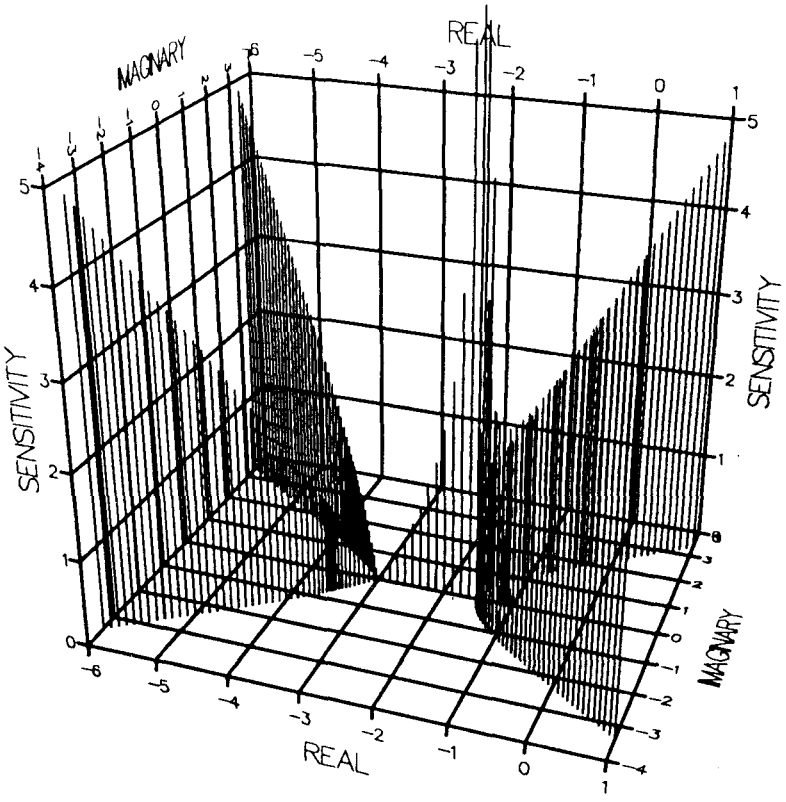
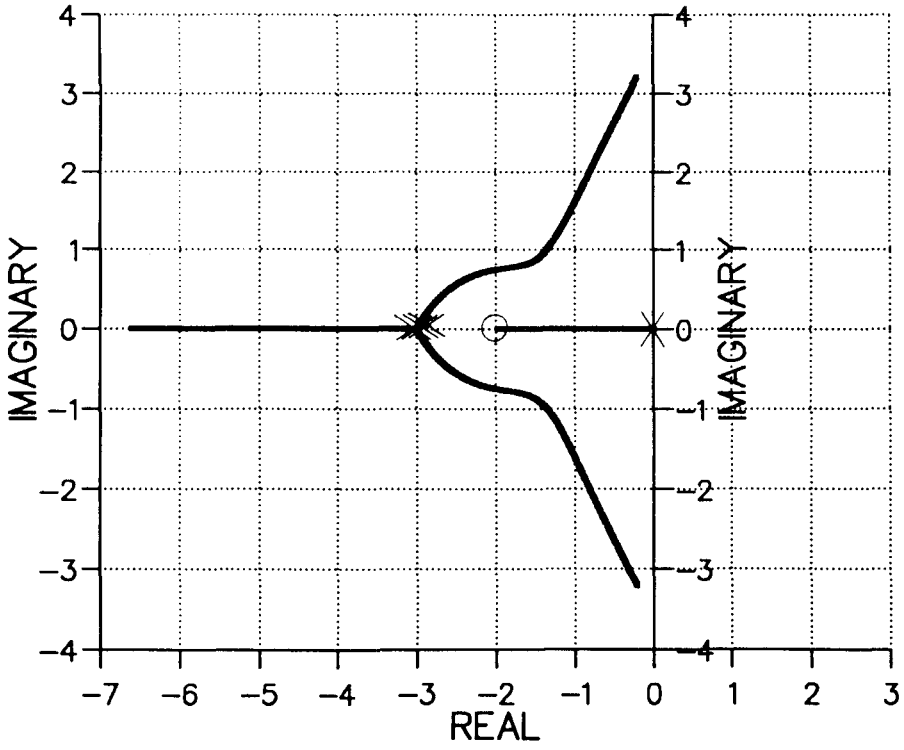


FIGURE 4.4 B FOURTH-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

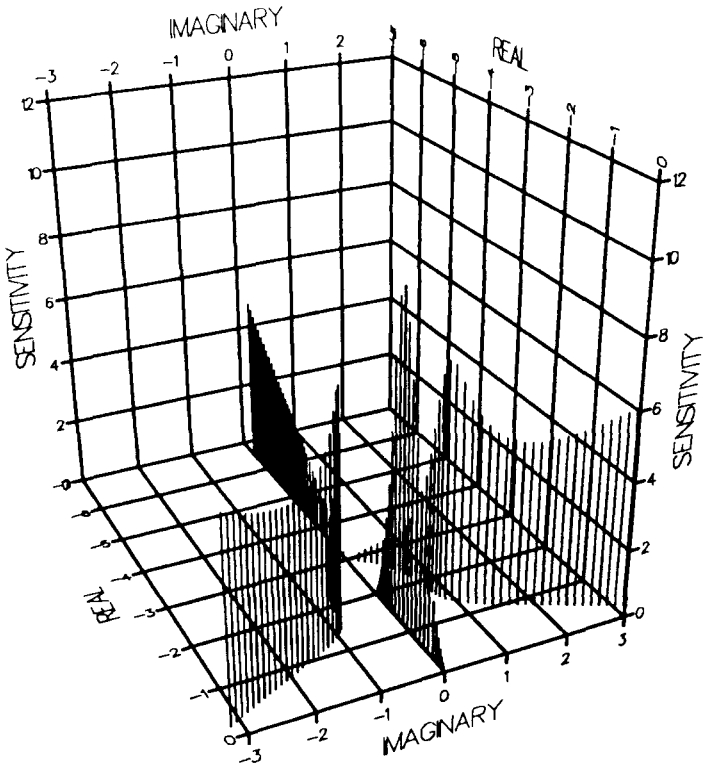
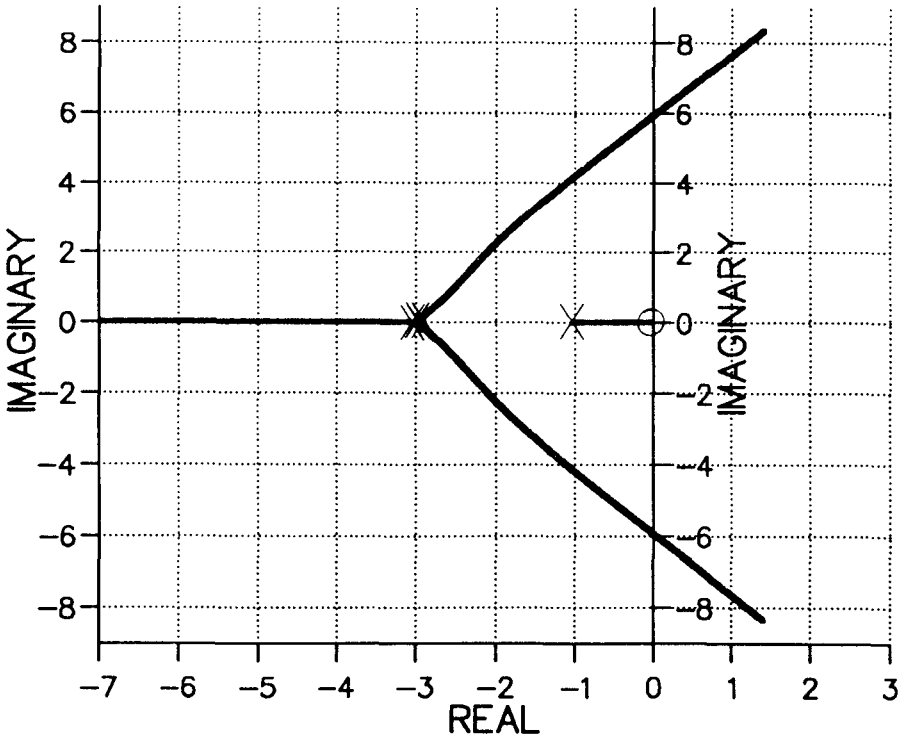


FIGURE 4.4 C FOURTH-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

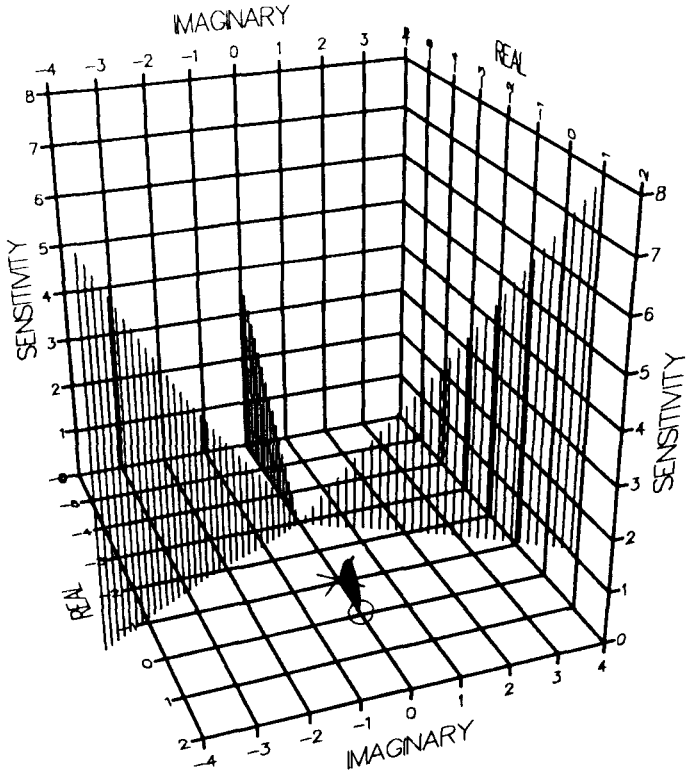
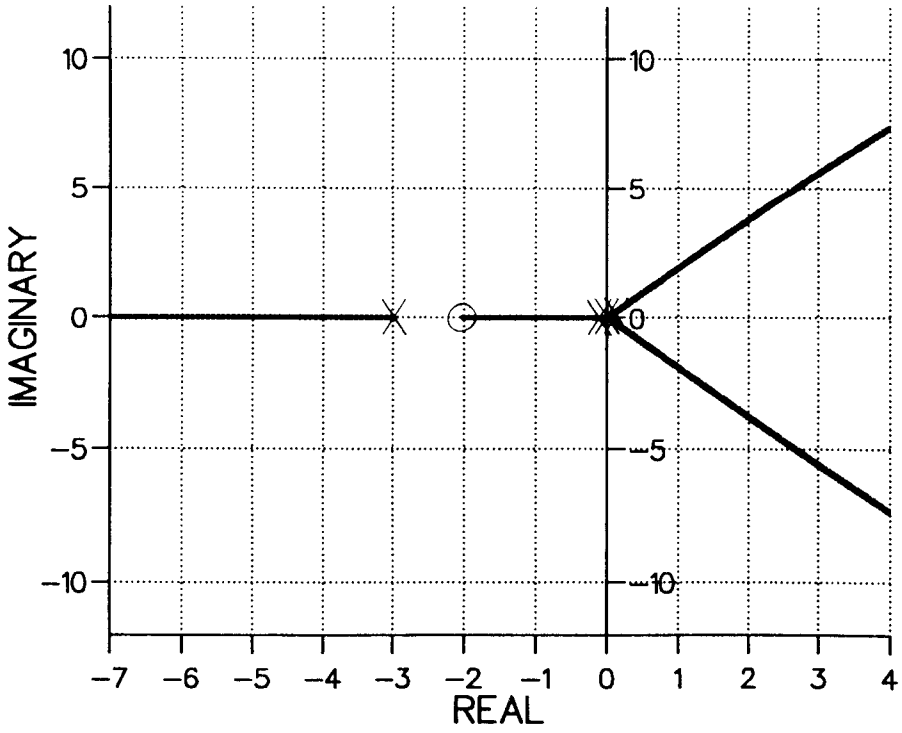


FIGURE 4.4 D FOURTH-ORDER PLANT

ROOT LOCUS



3 D SENSITIVITY PROFILE

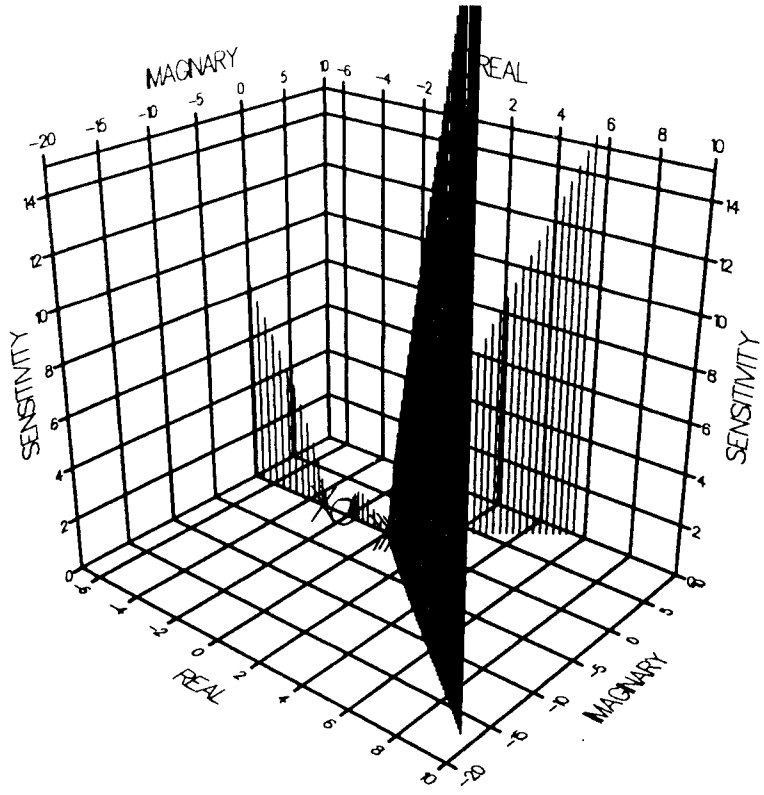
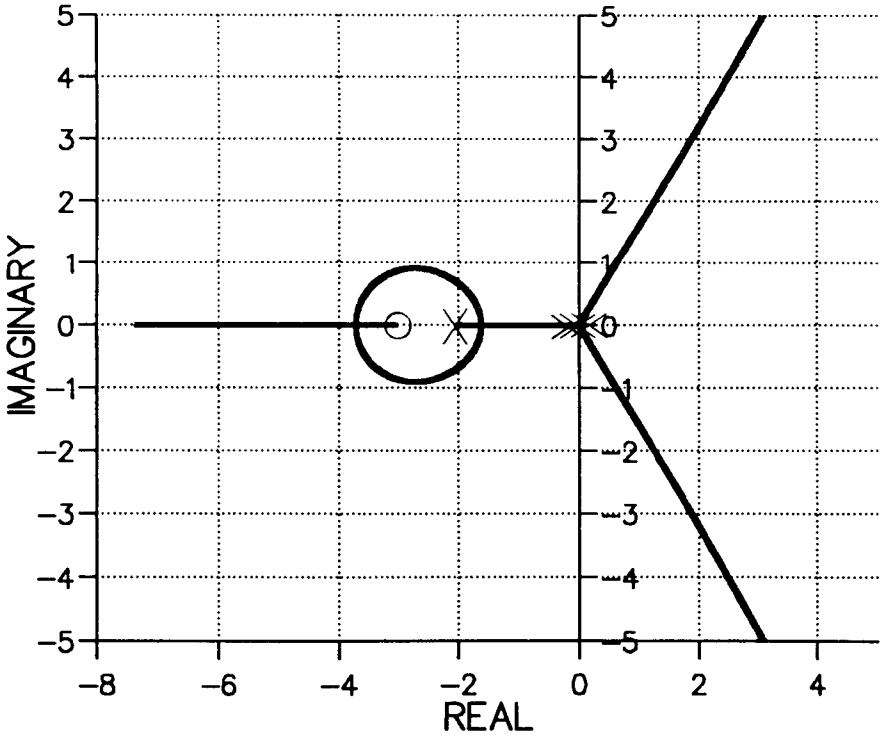


FIGURE 4.4 E FOURTH-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

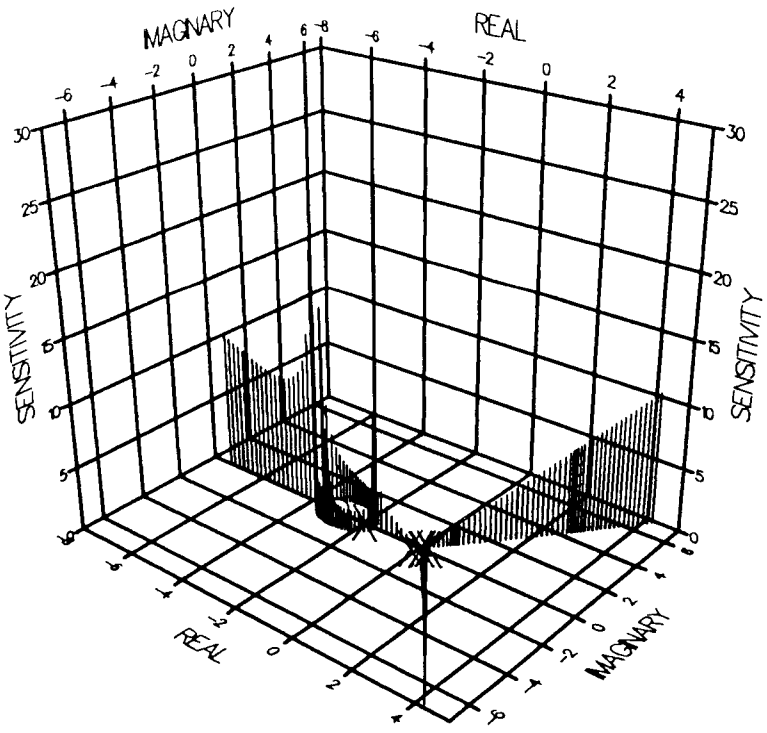
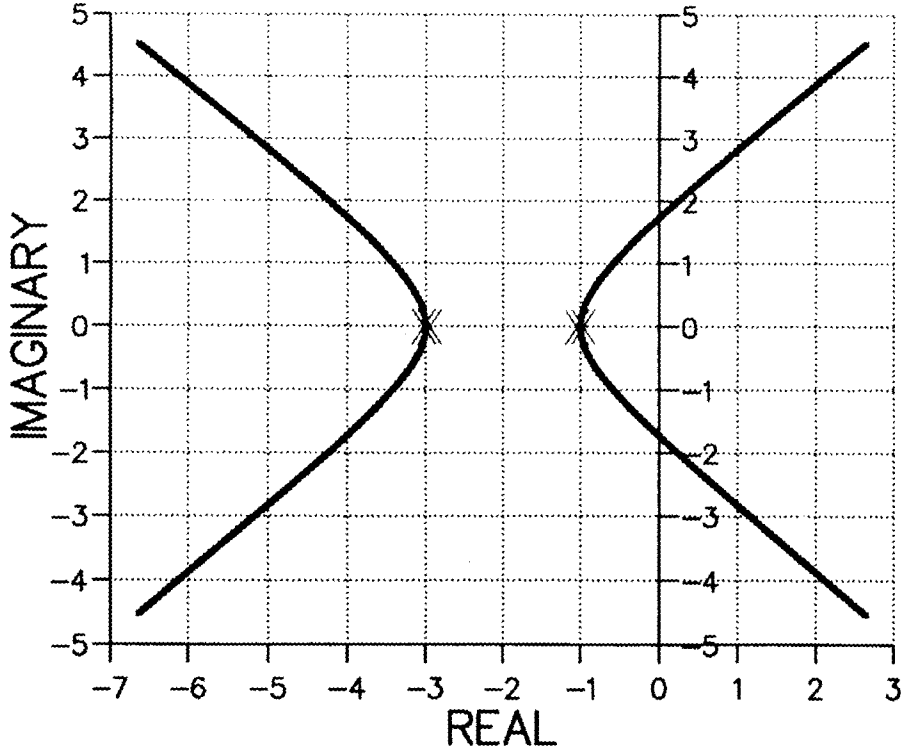


FIGURE 4.4 F FOURTH-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

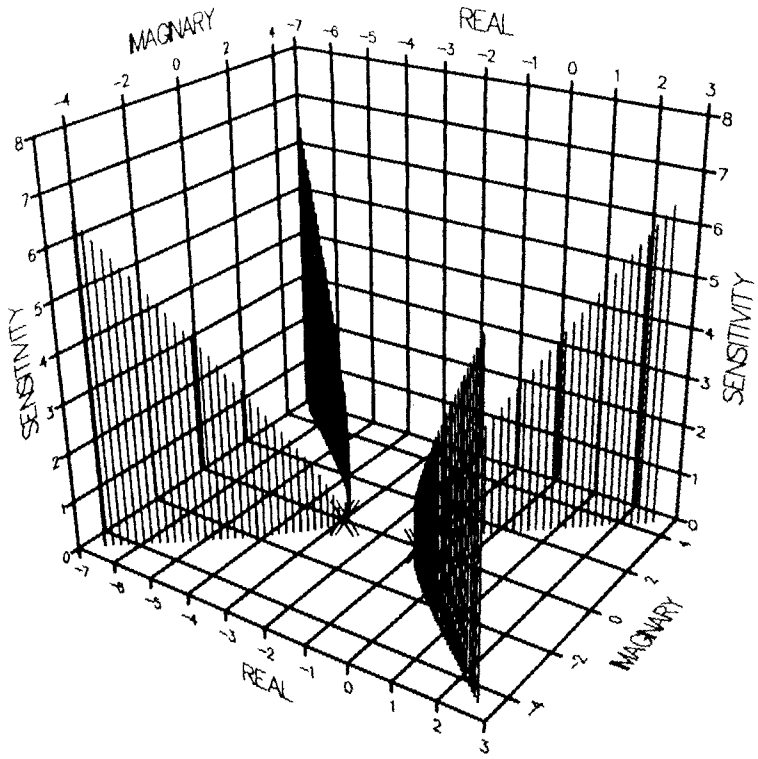


FIGURE 4.4 G FOURTH-ORDER PLANT

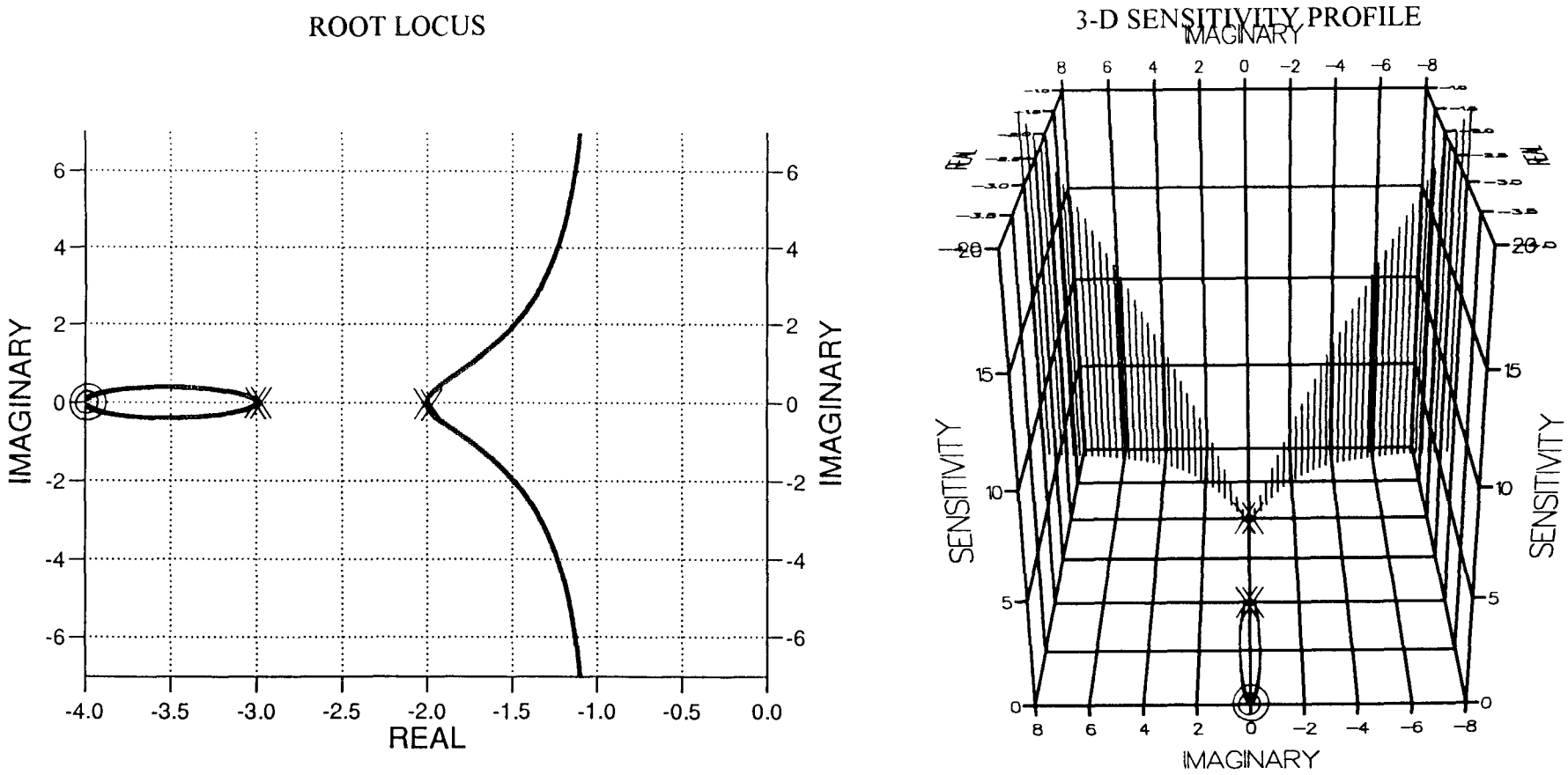
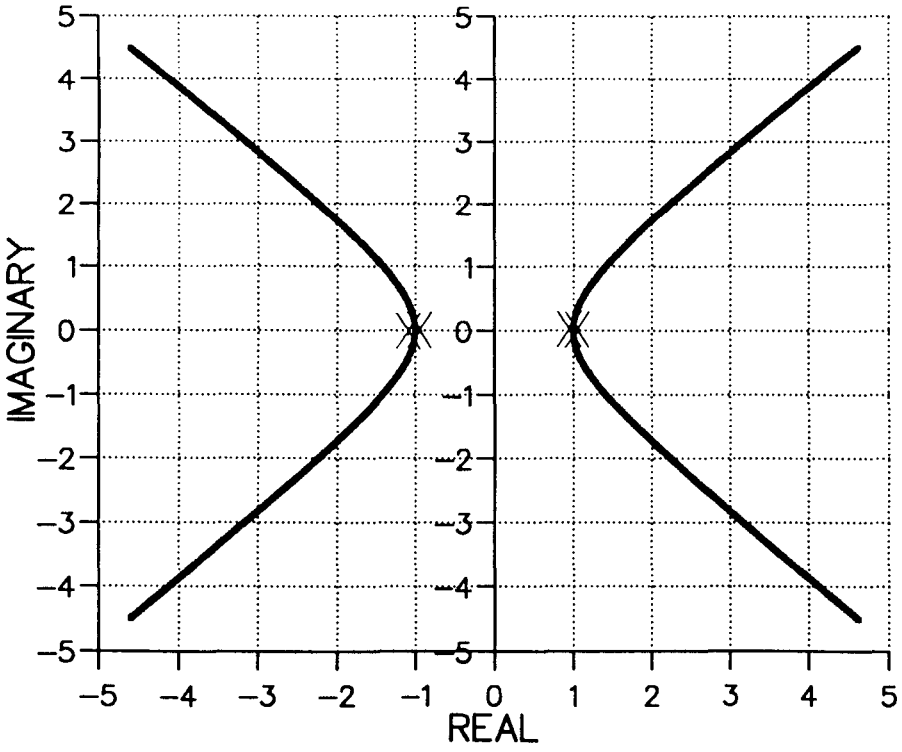


FIGURE 4.4 H FOURTH-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

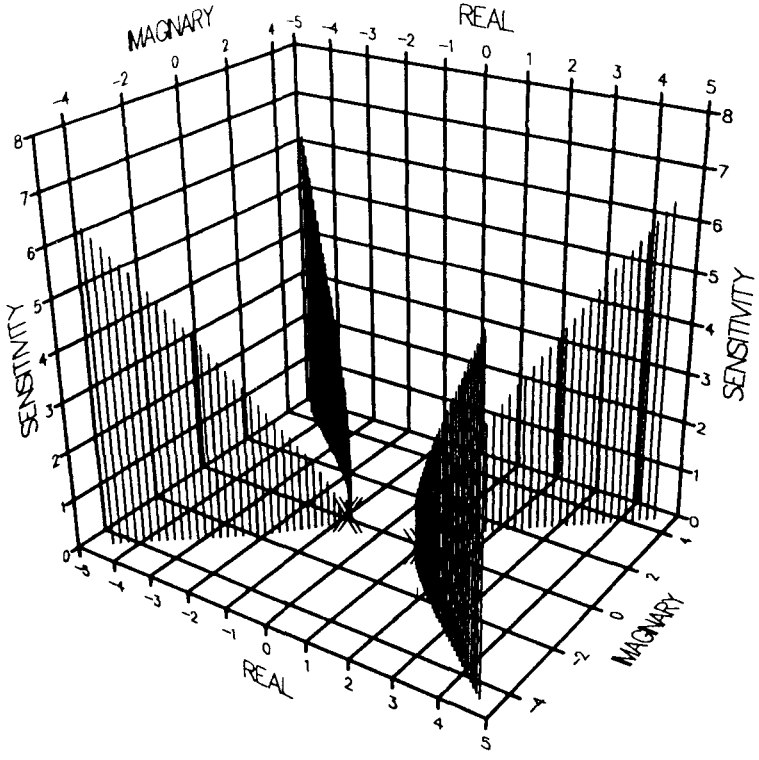
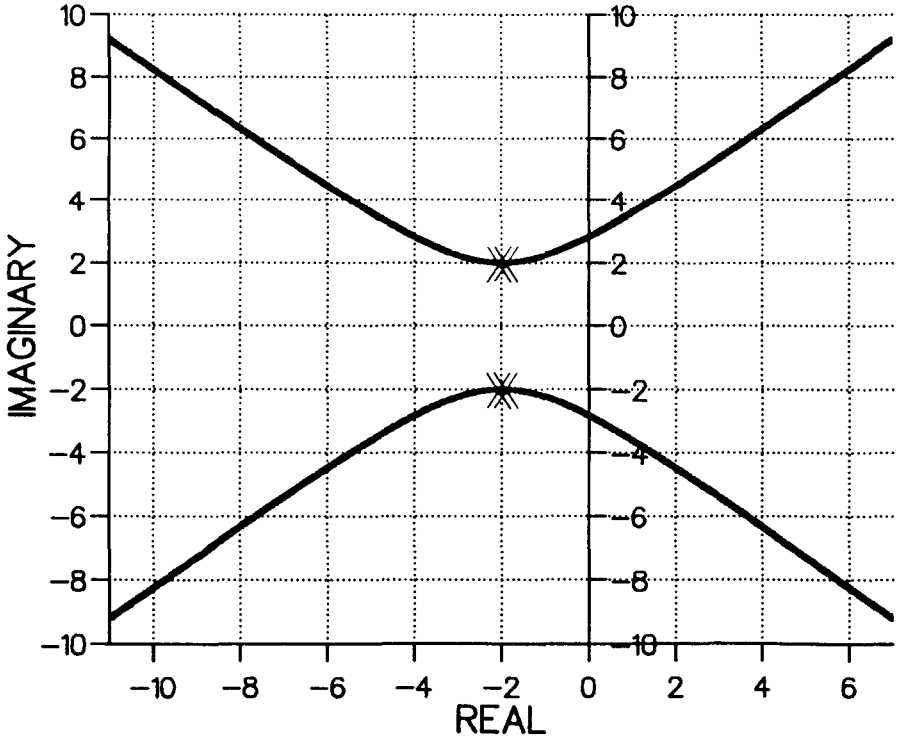


FIGURE 4.4 I FOURTH-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

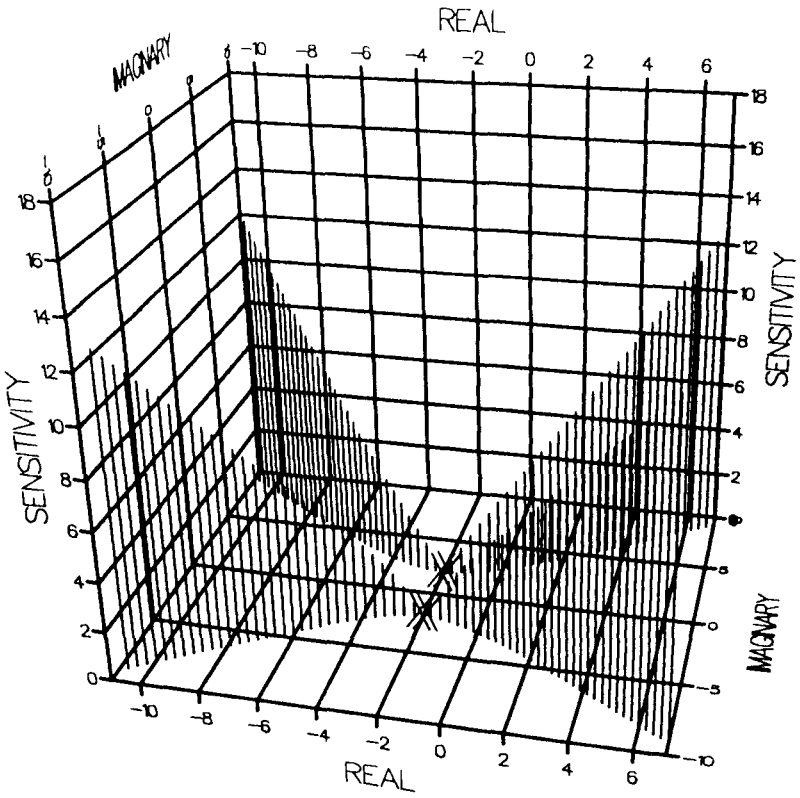
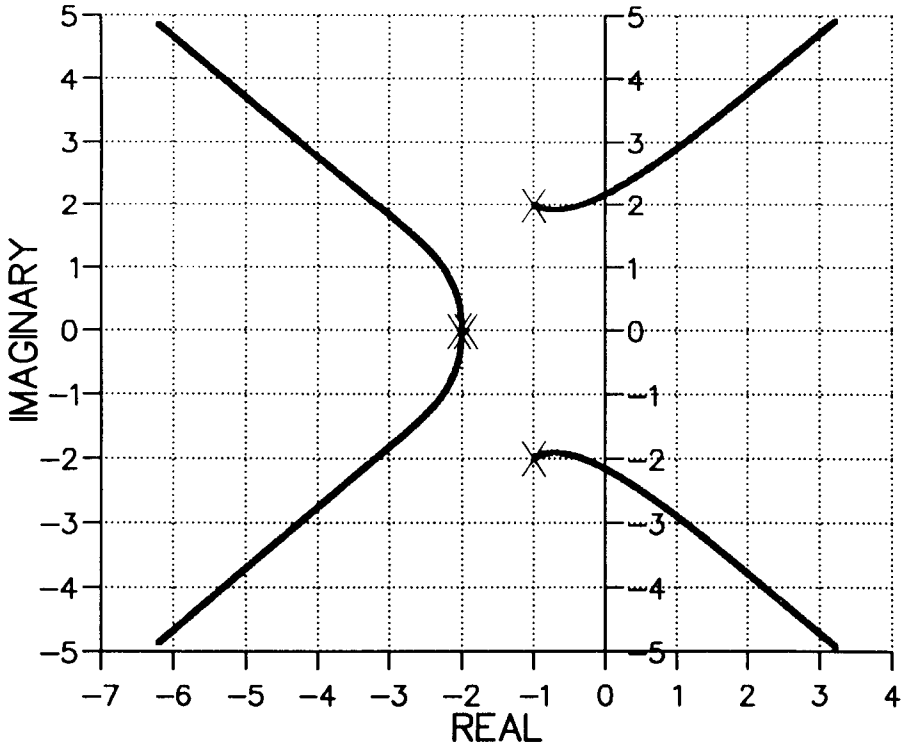


FIGURE 4.4 J FOURTH-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

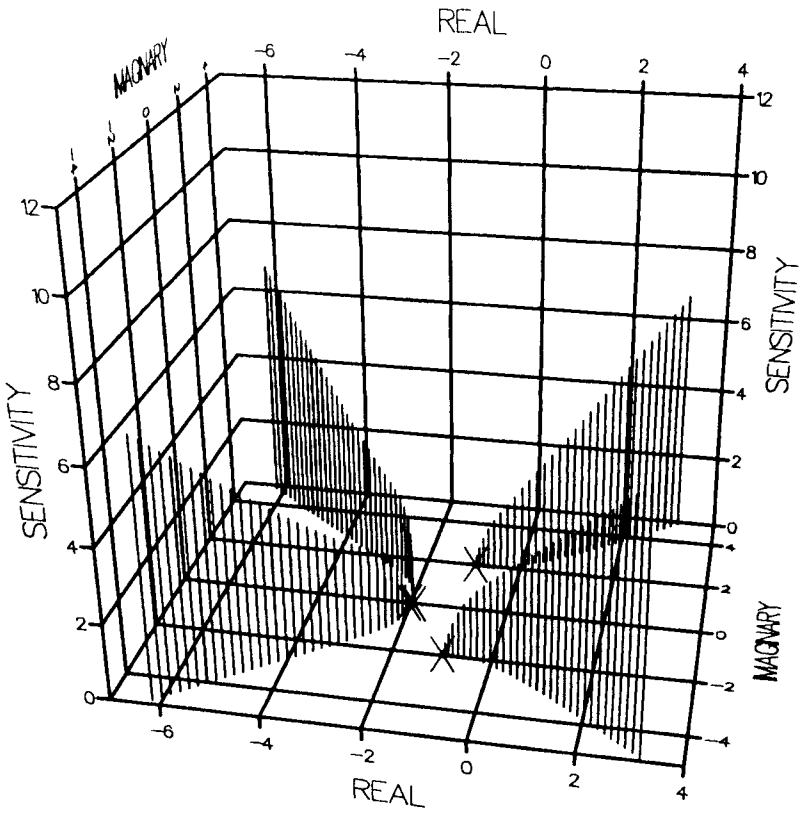
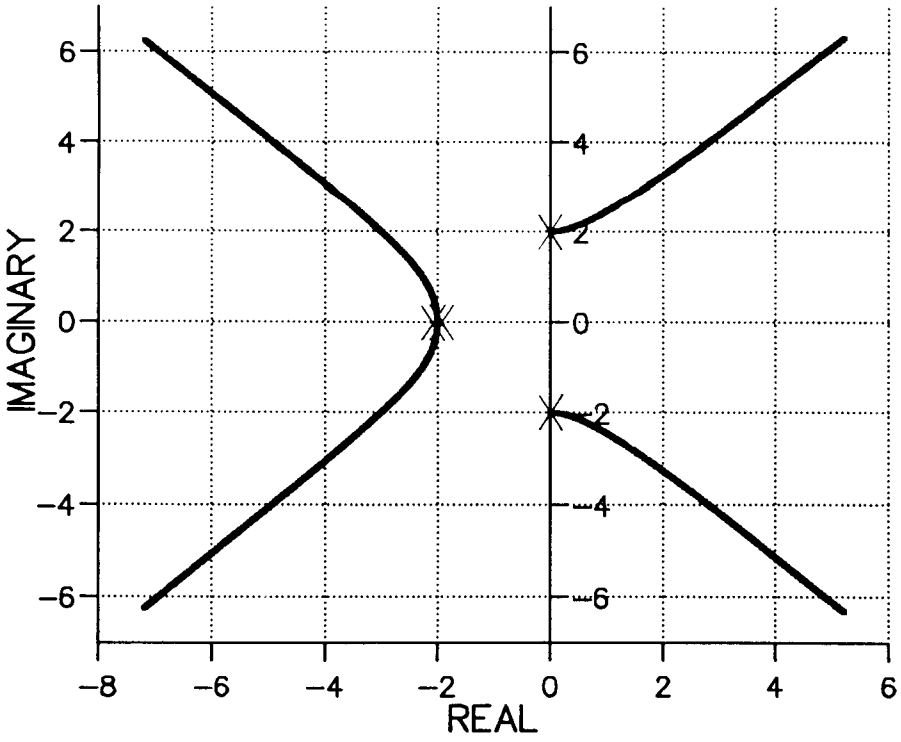


FIGURE 4.4 K FOURTH-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

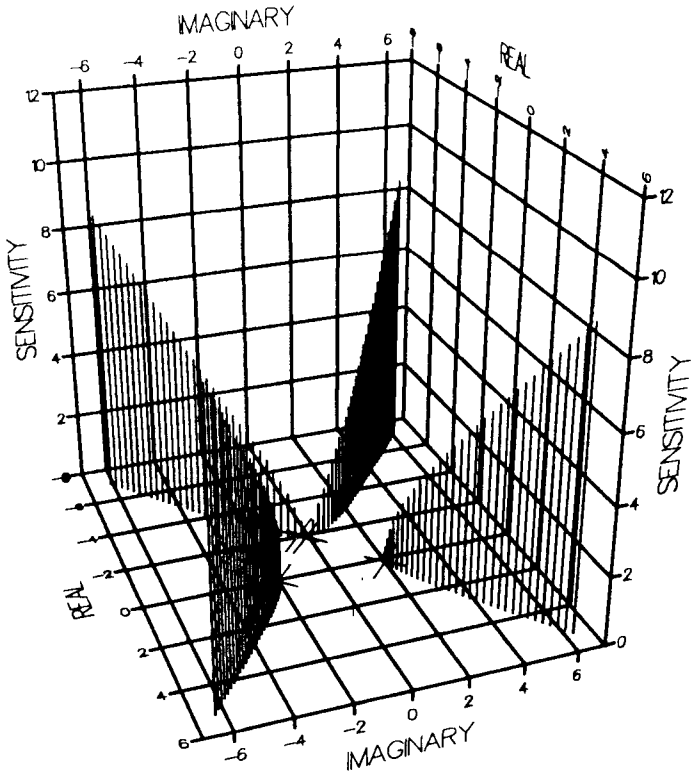
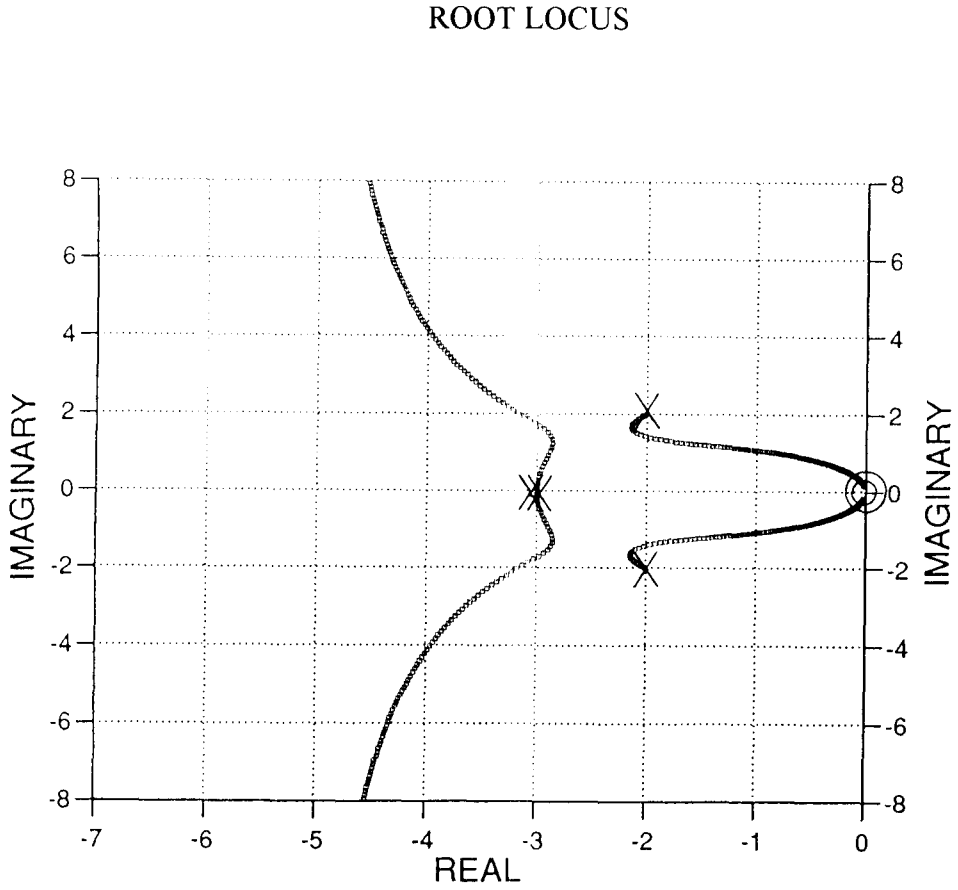


FIGURE 4.4 L FOURTH-ORDER PLANT



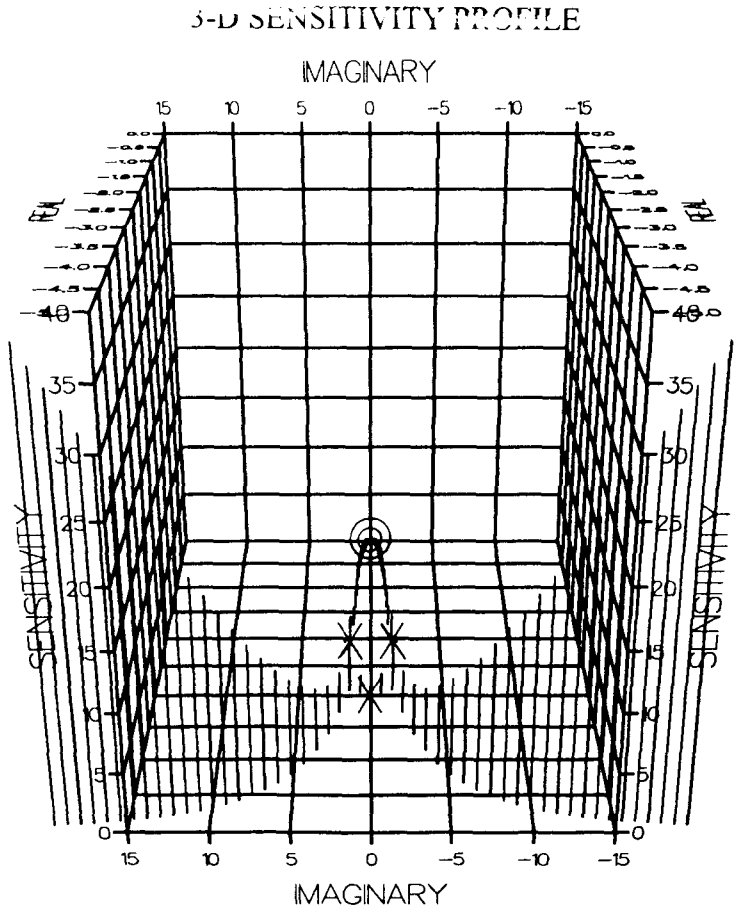


FIGURE 4.4 M FOURTH-ORDER PLANT

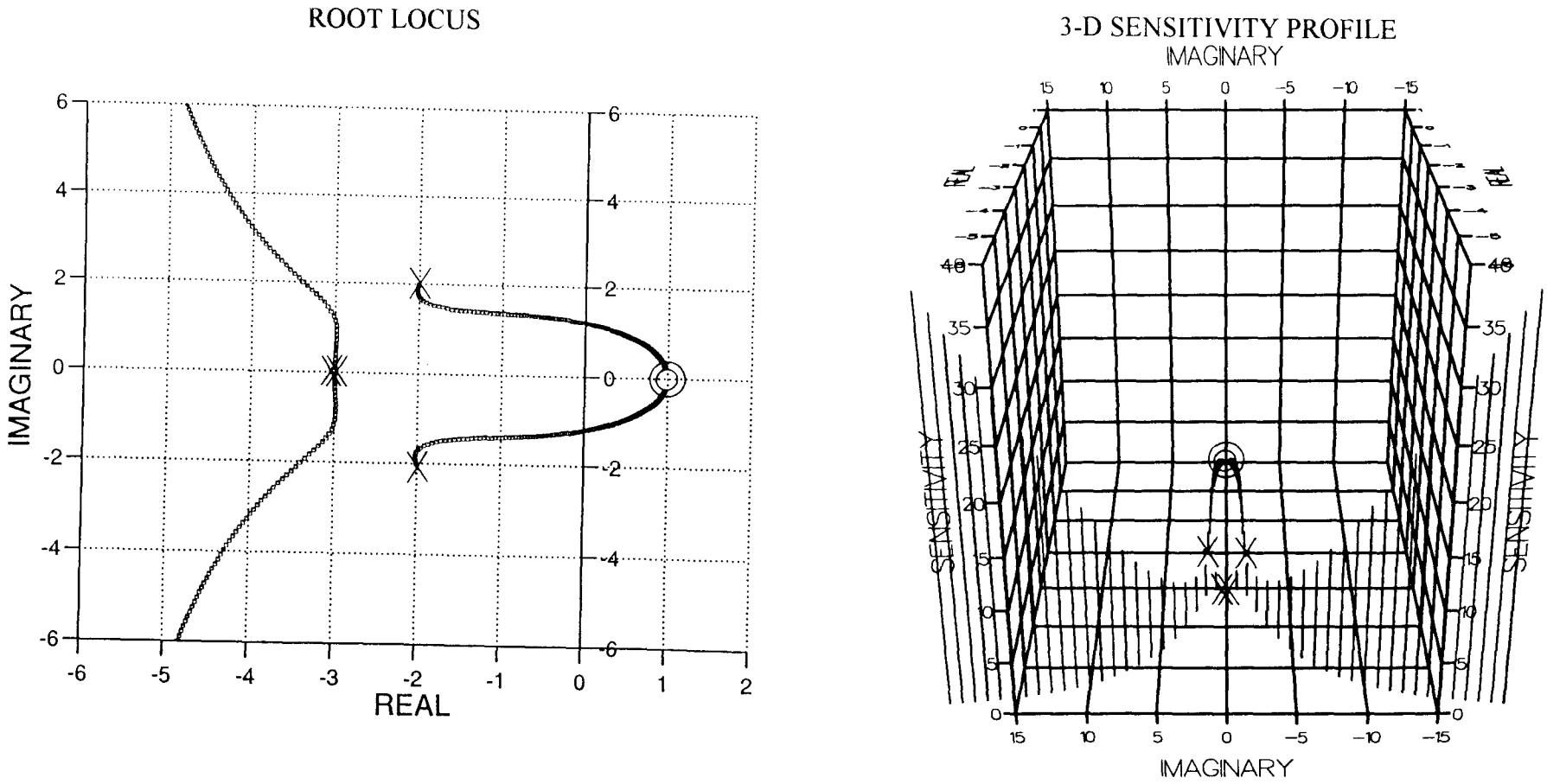
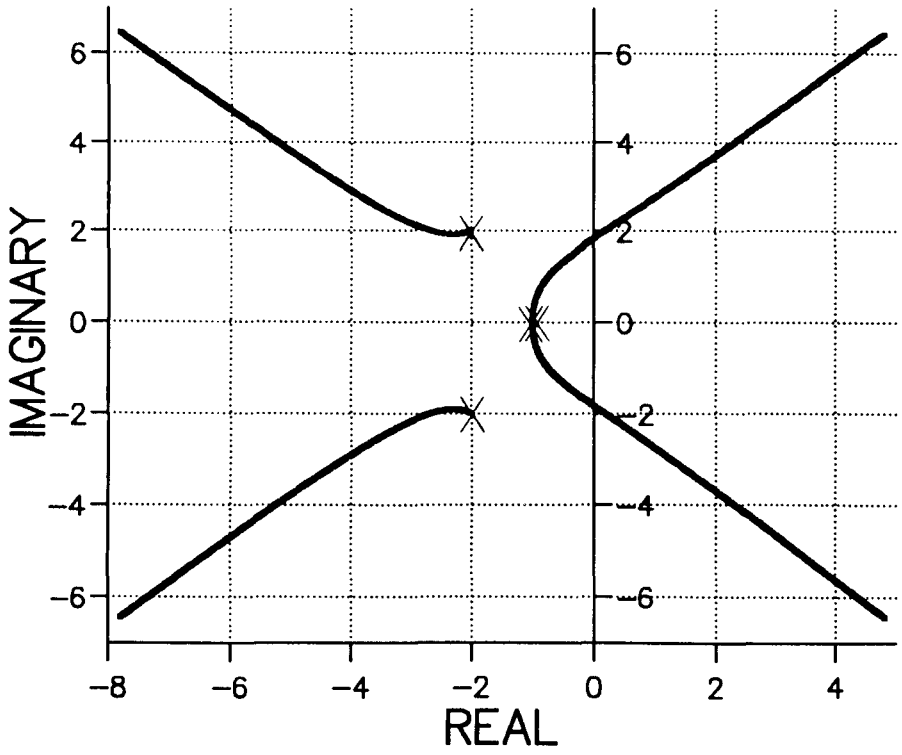


FIGURE 4.4 N FOURTH-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

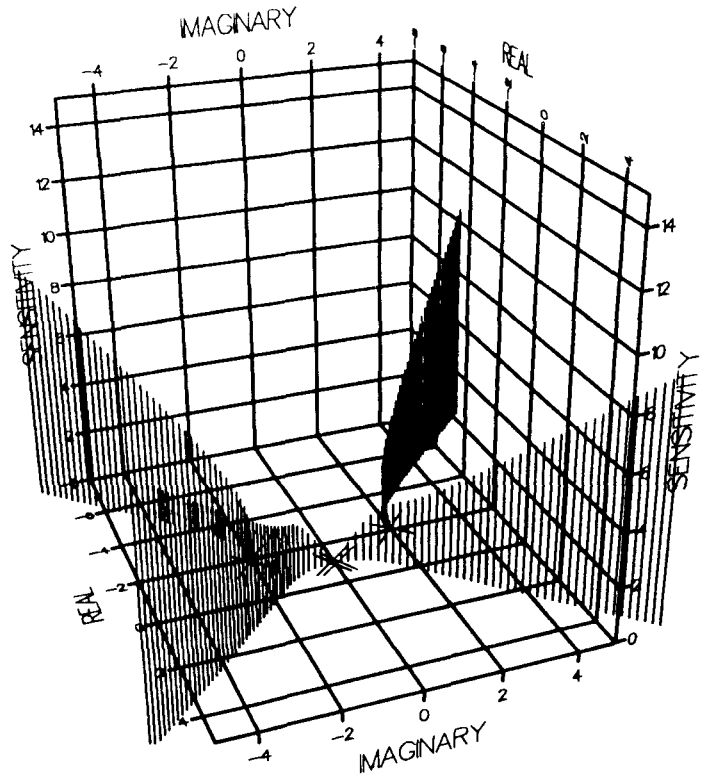
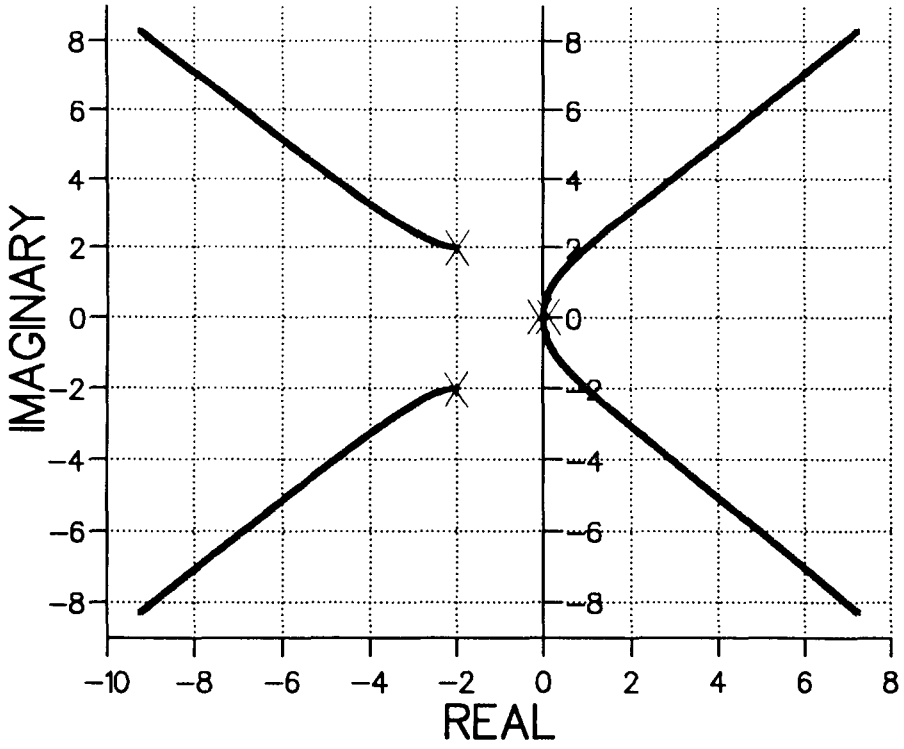


FIGURE 4.4 O FOURTH-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

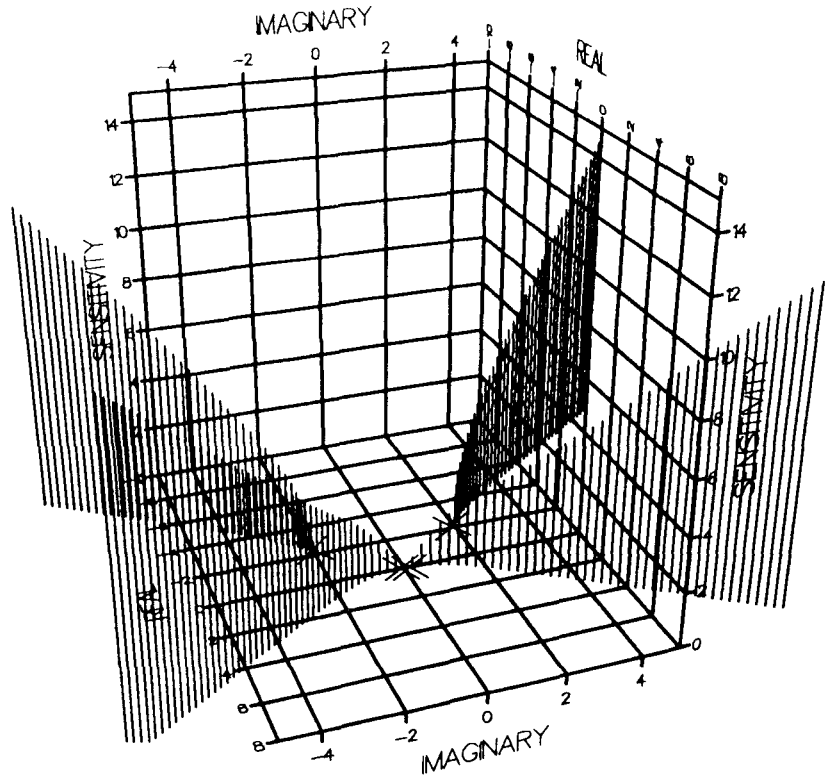


FIGURE 4.4 P FOURTH-ORDER PLANT

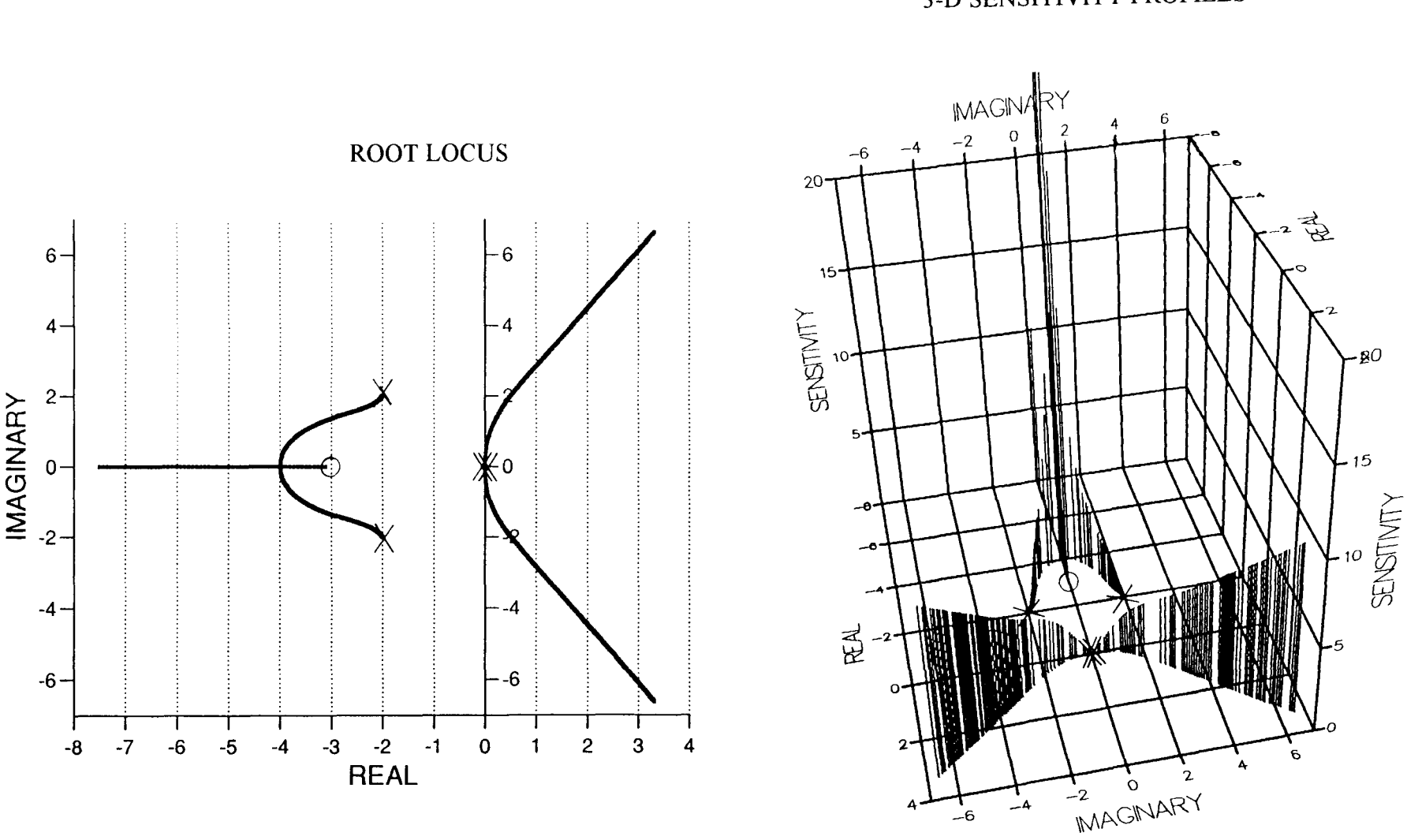
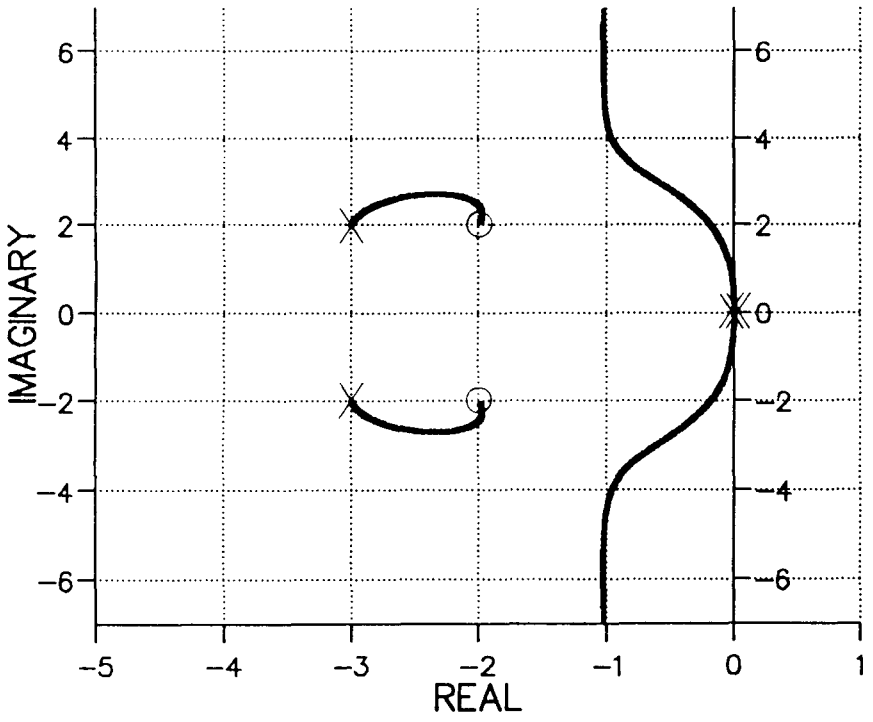


FIGURE 4.4 Q FOURTH-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

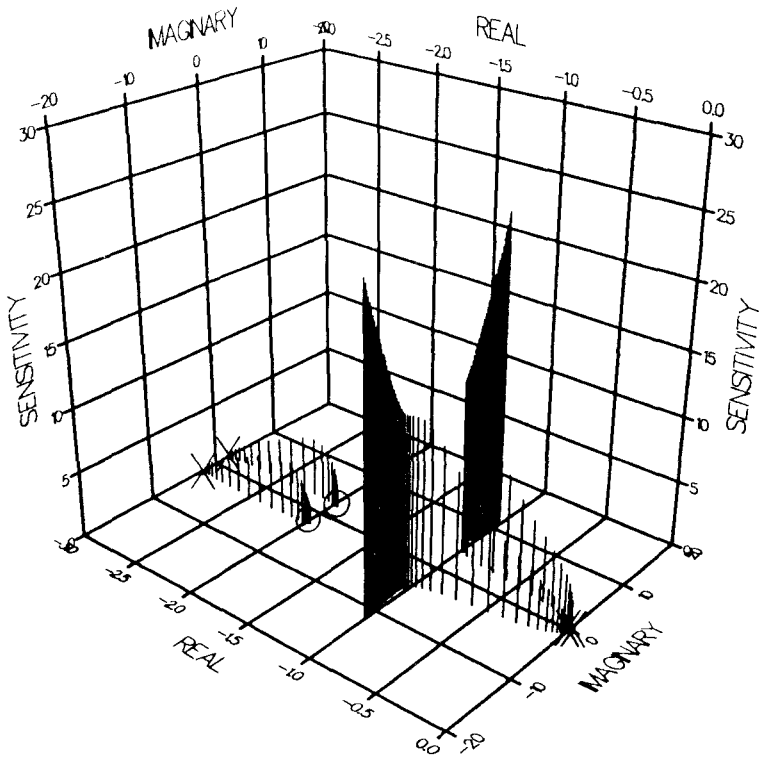


FIGURE 4.4 R FOURTH-ORDER PLANT

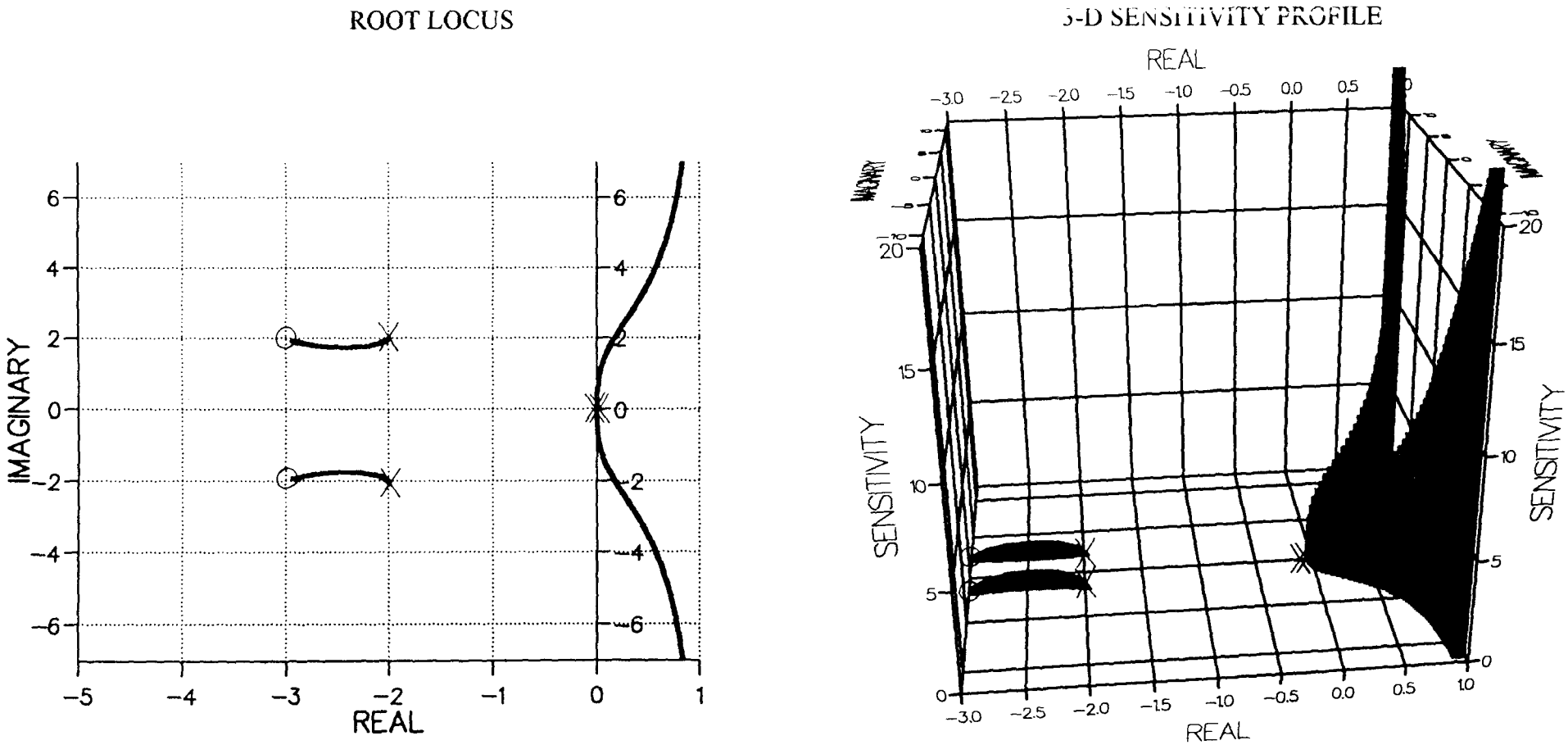
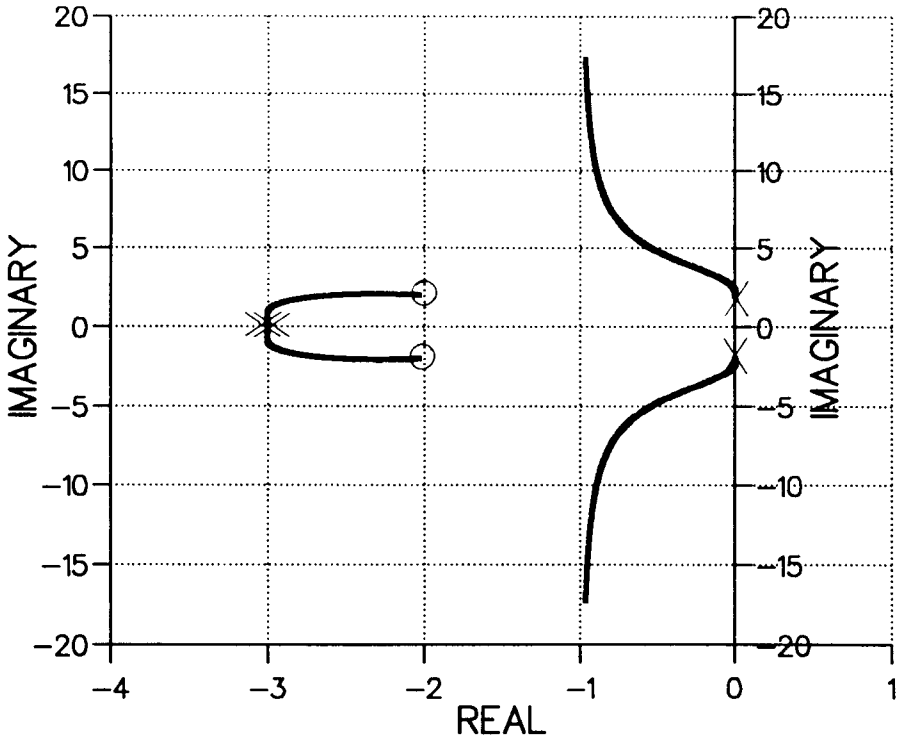


FIGURE 4.4 S FOURTH-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

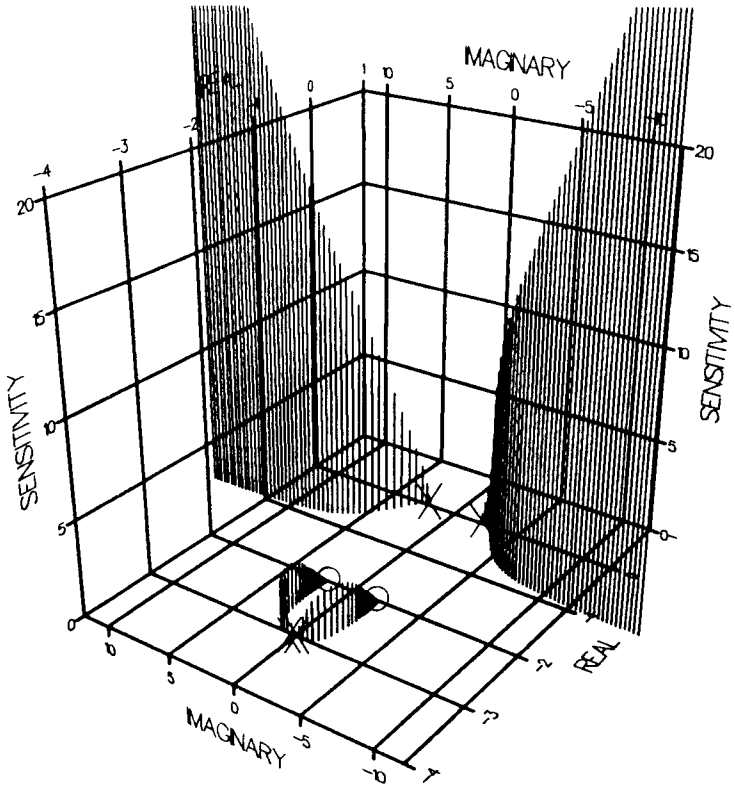
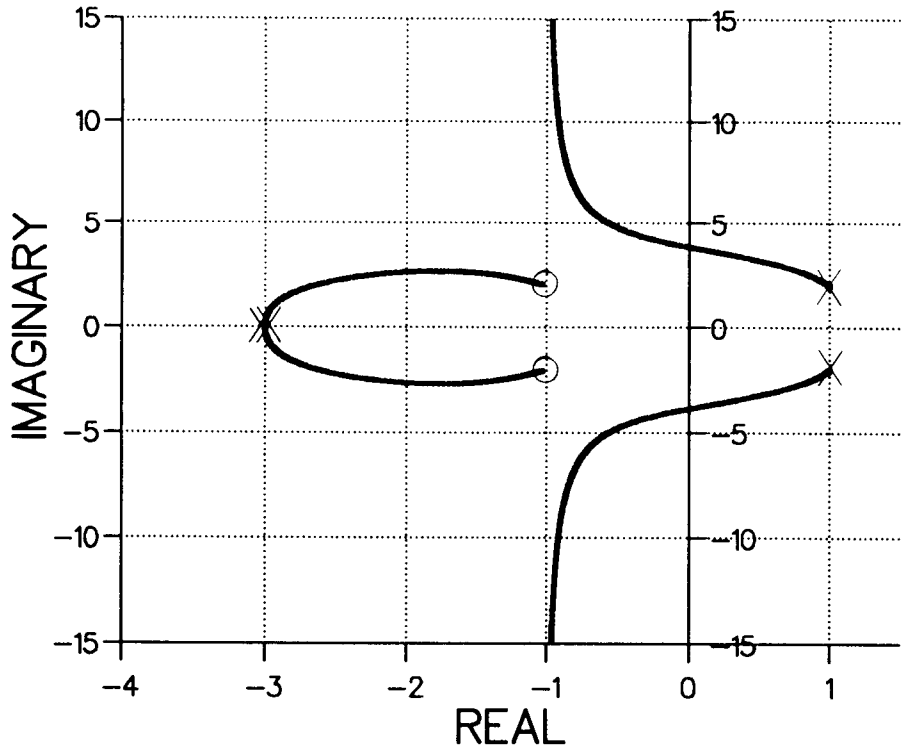


FIGURE 4.4 T FOURTH-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

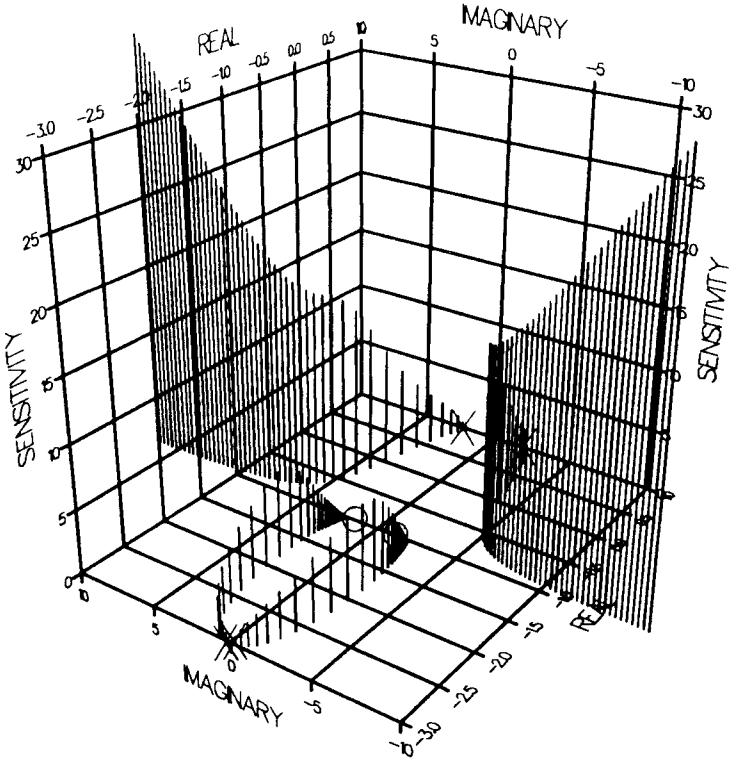
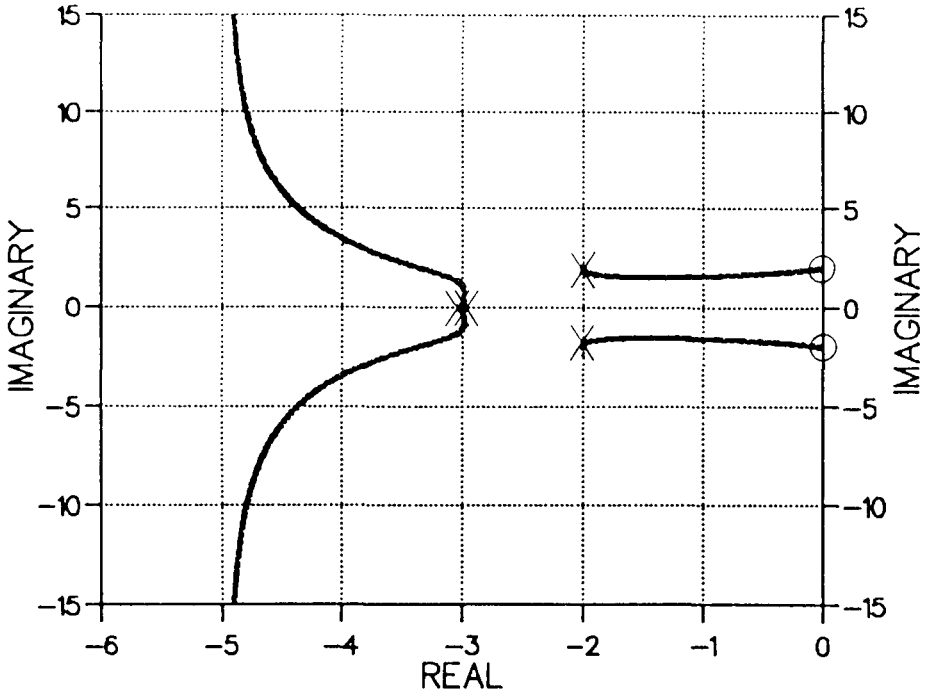


FIGURE 4.4 U FOURTH-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

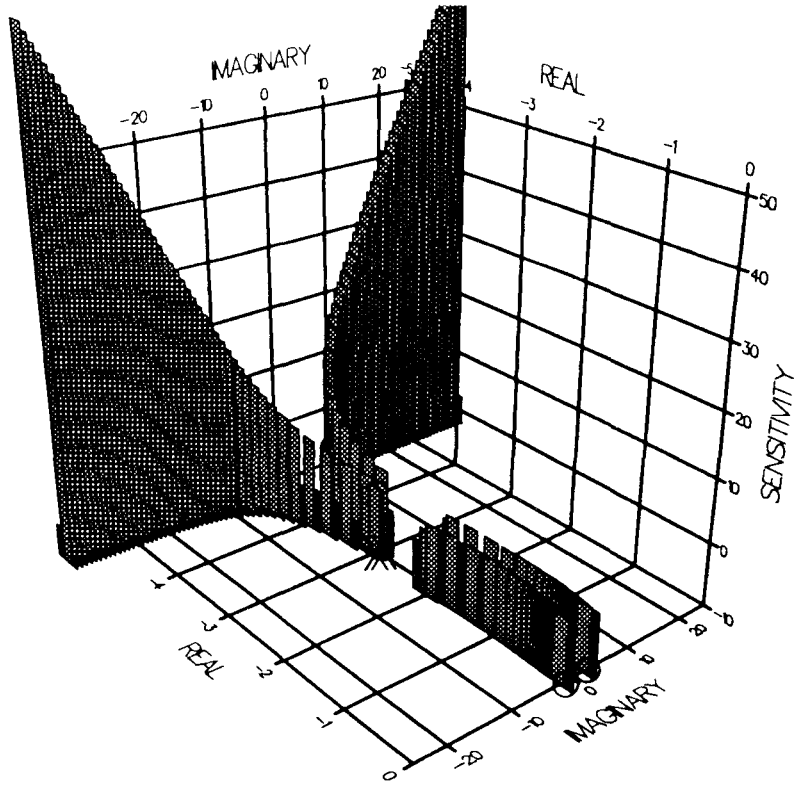
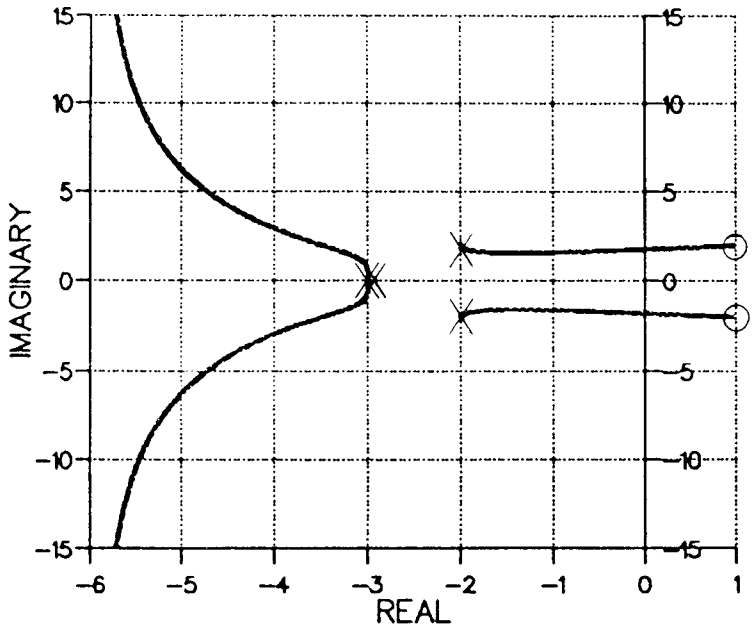


FIGURE 4.4 V FOURTH-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

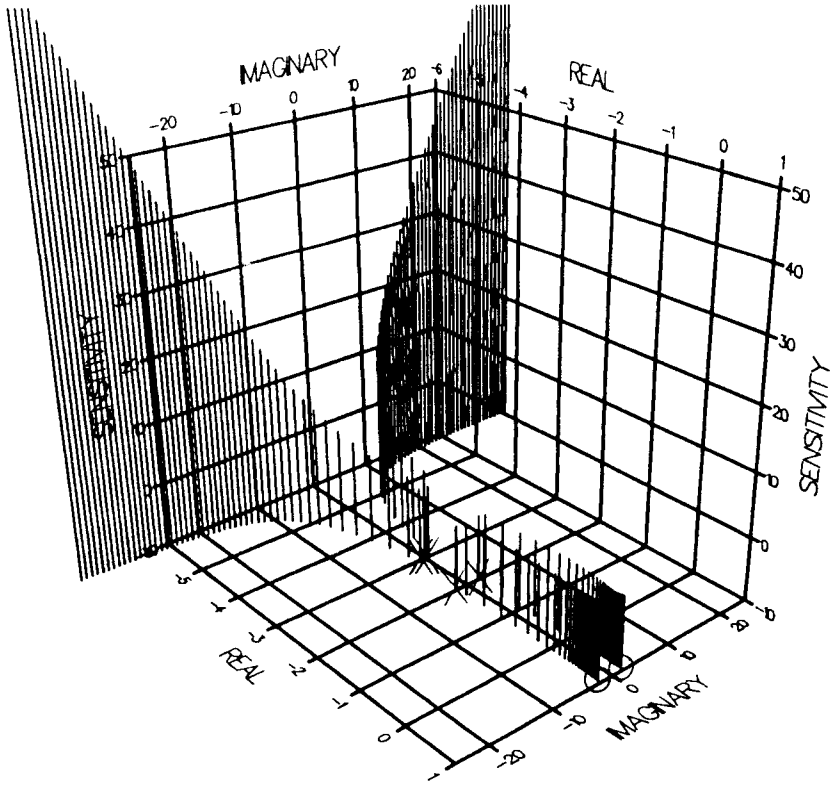
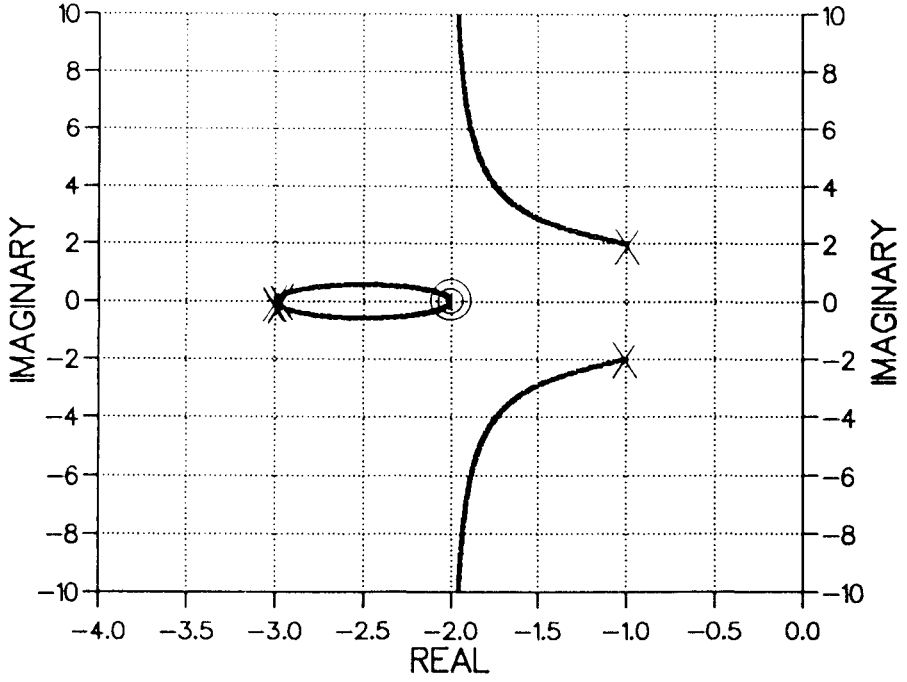


FIGURE 4.4 W FOURTH-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

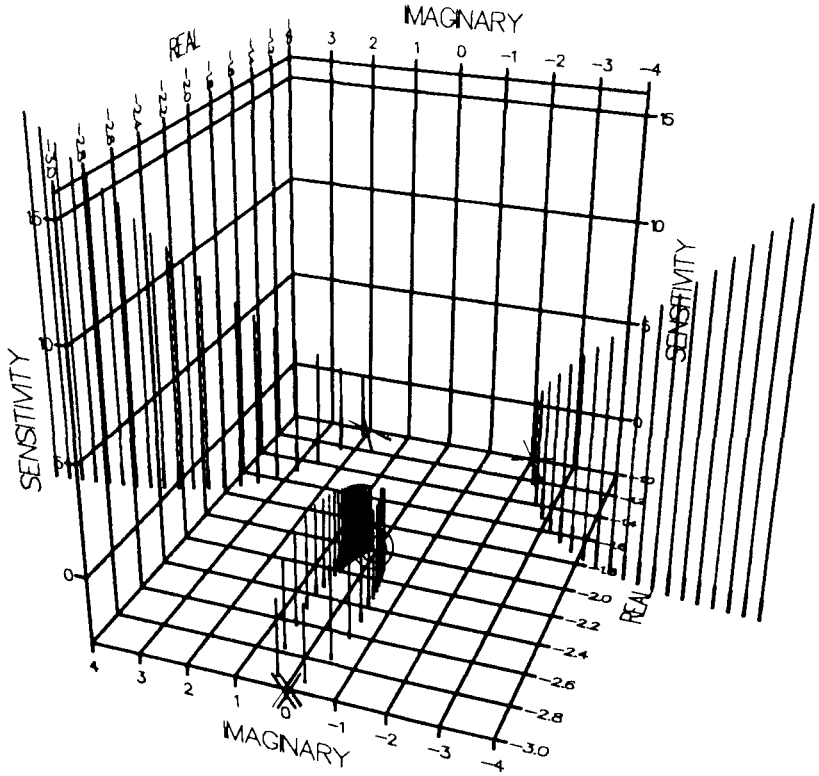


FIGURE 4.4 X FOURTH-ORDER PLANT

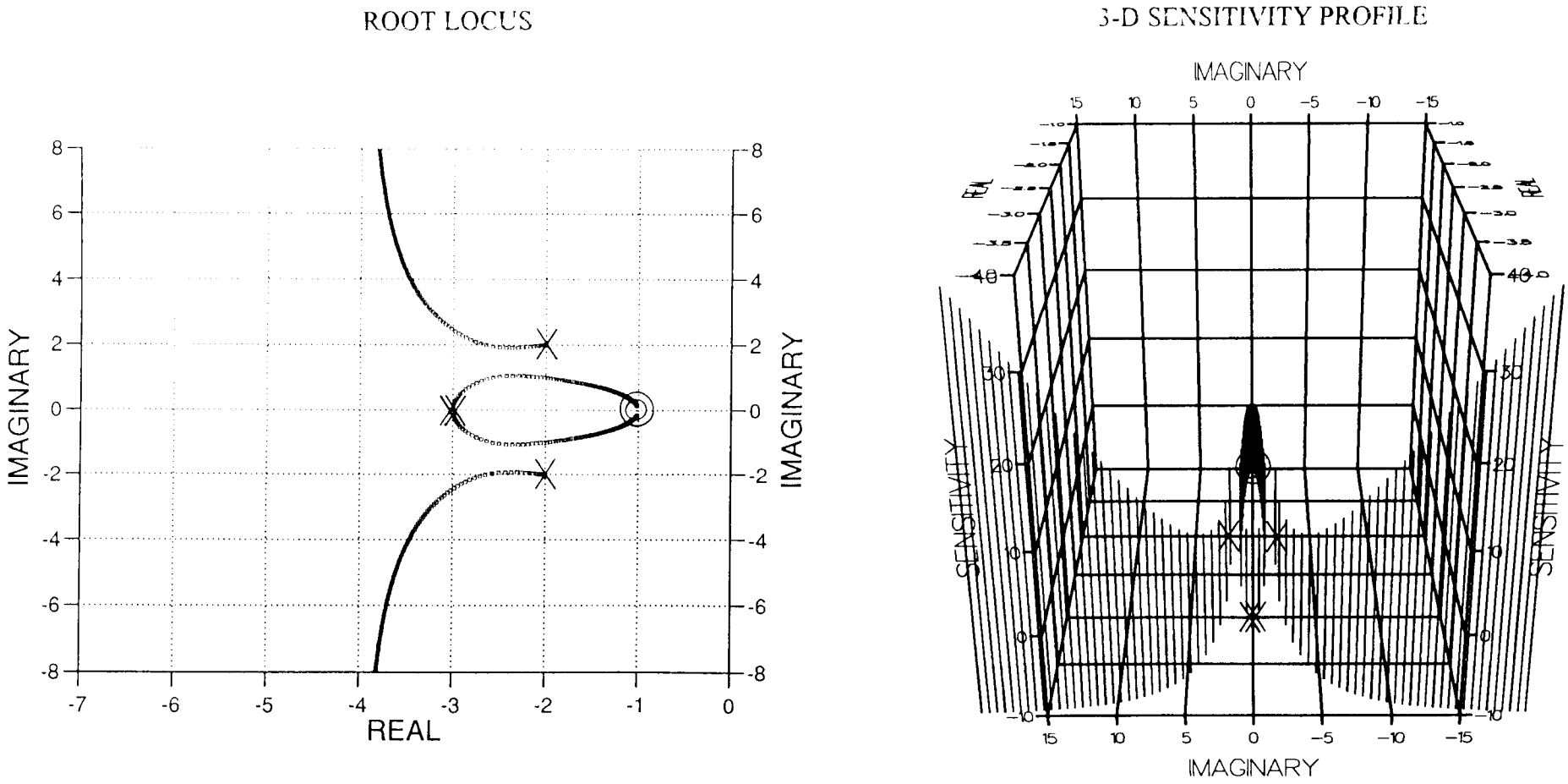
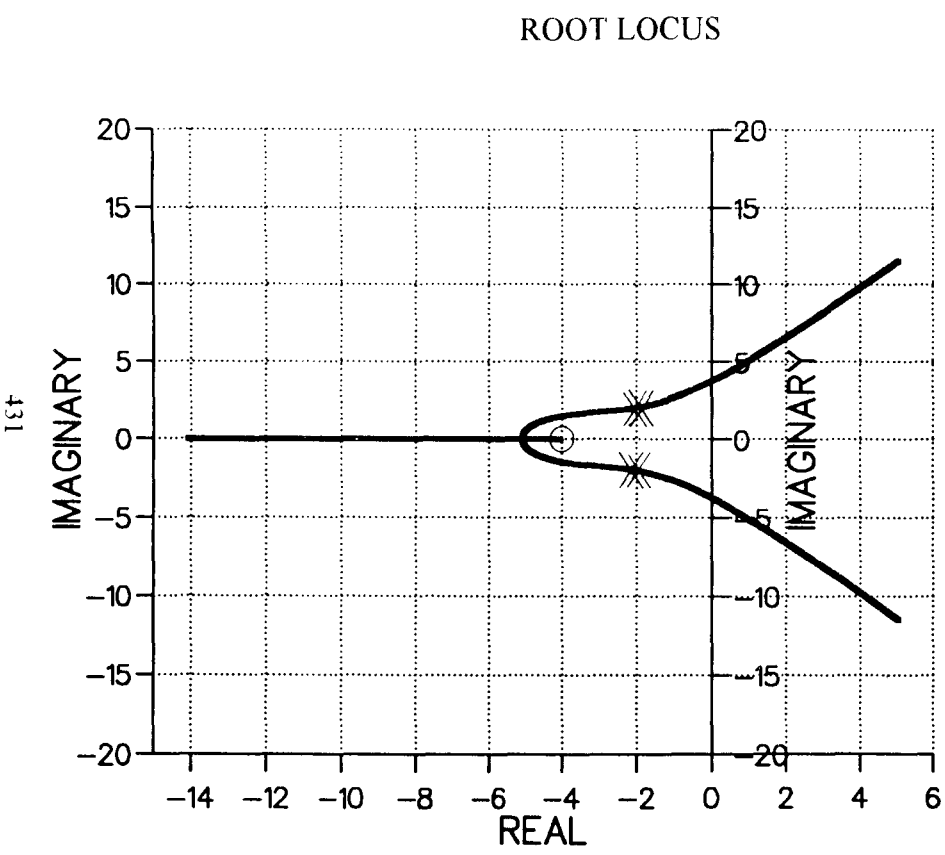


FIGURE 4.4 Y FOURTH-ORDER PLANT



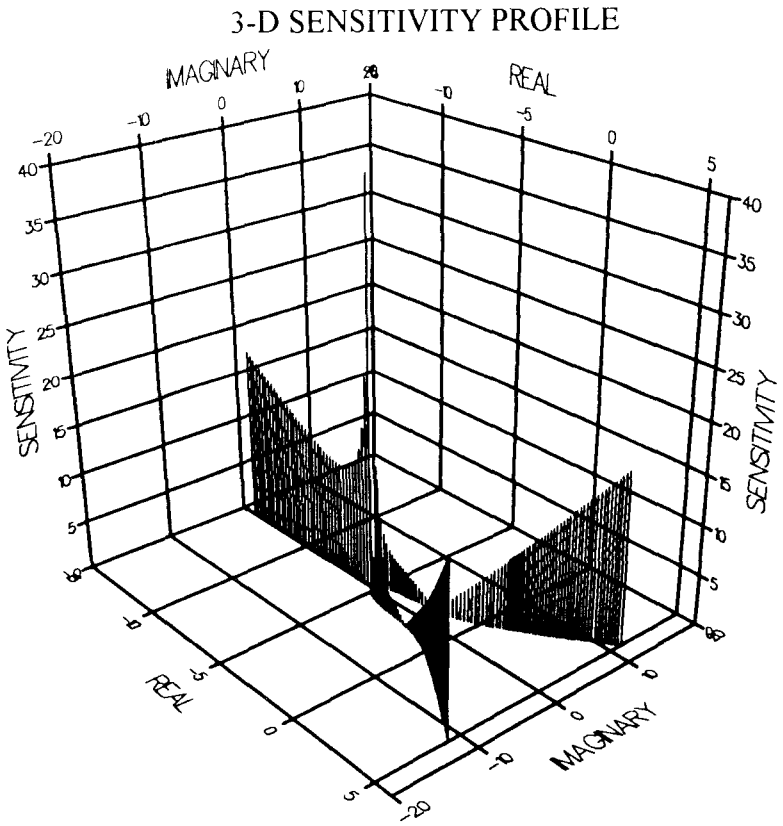


FIGURE 4.4 Z FOURTH-ORDER PLANT

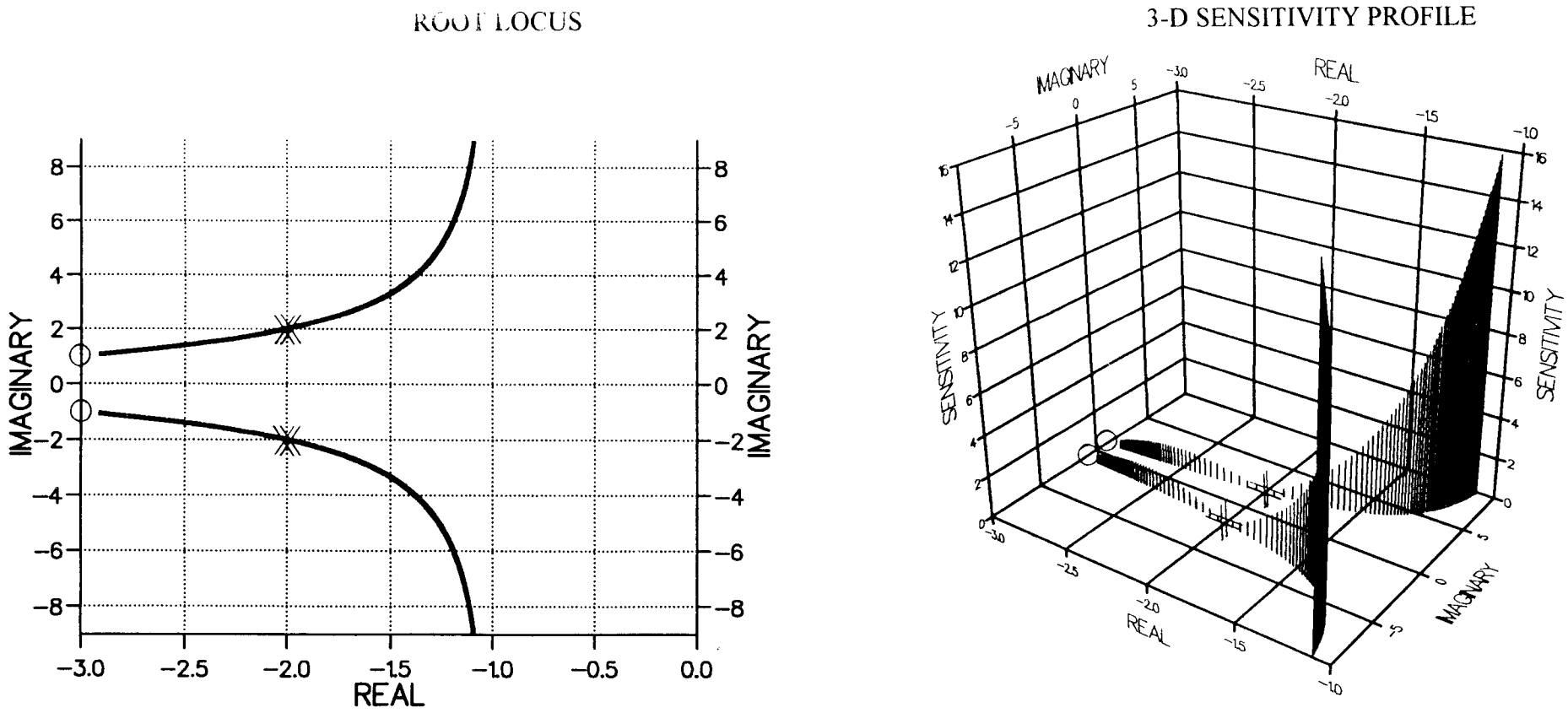
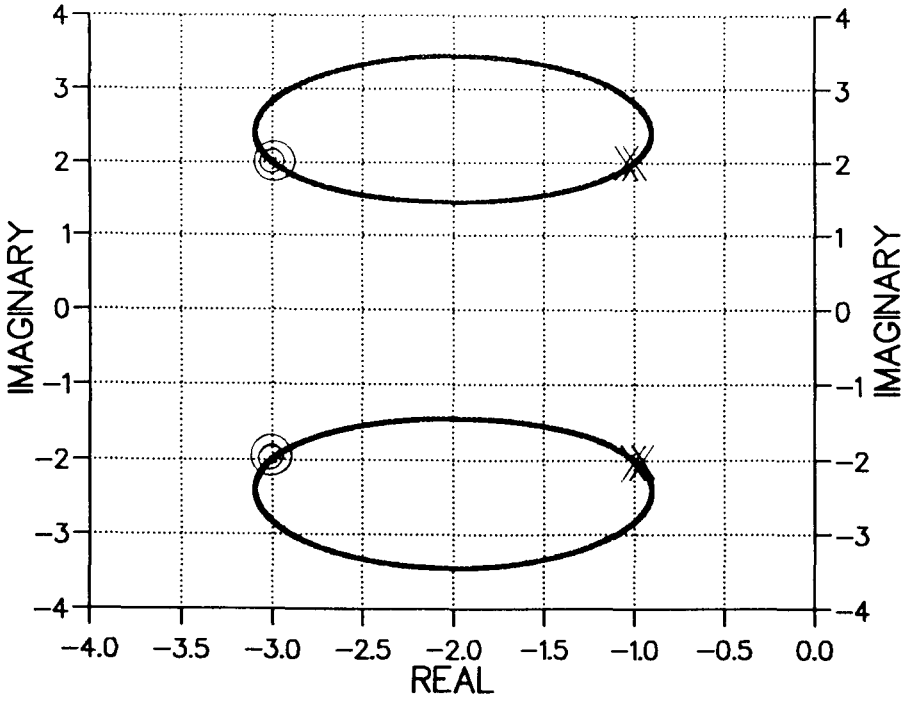


FIGURE 4.4 #A FOURTH-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

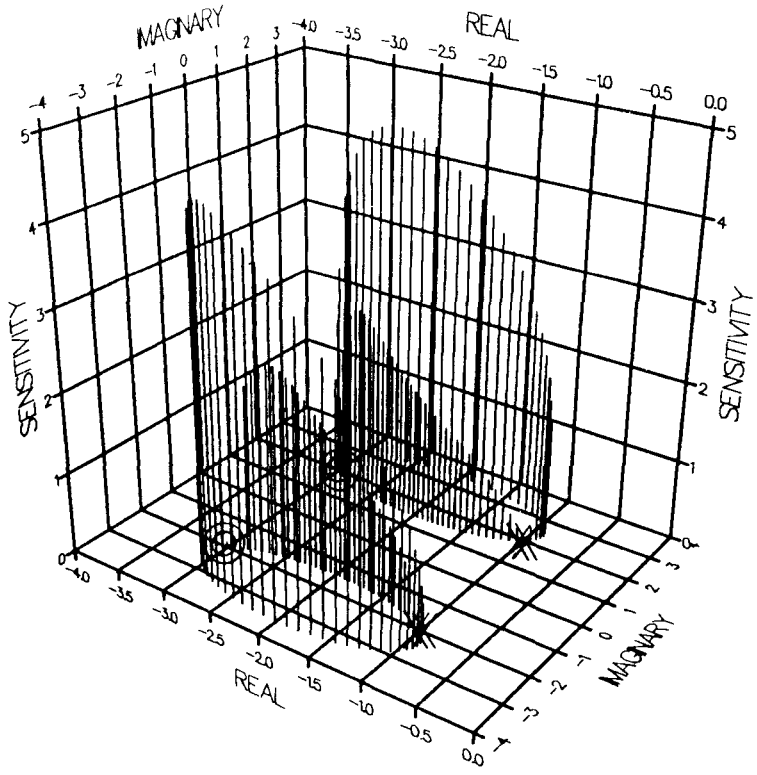
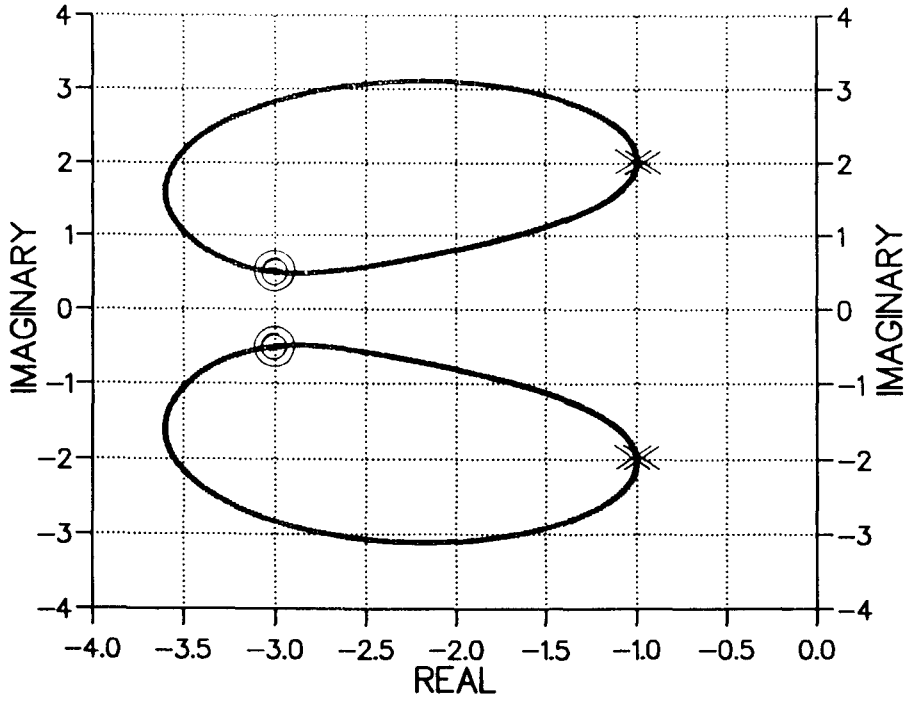


FIGURE 4.4 #B FOURTH-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

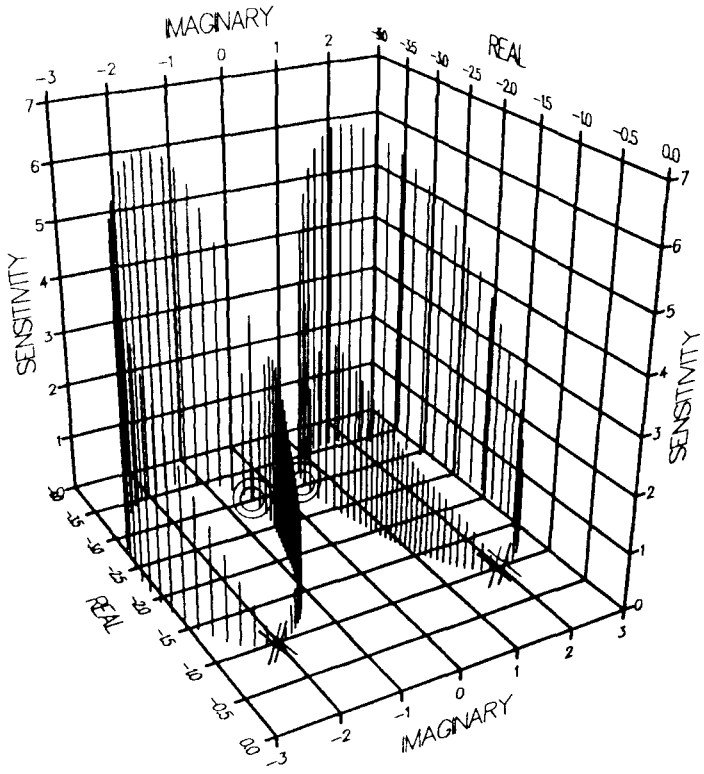


FIGURE 4.4 #C FOURTH-ORDER PLANT

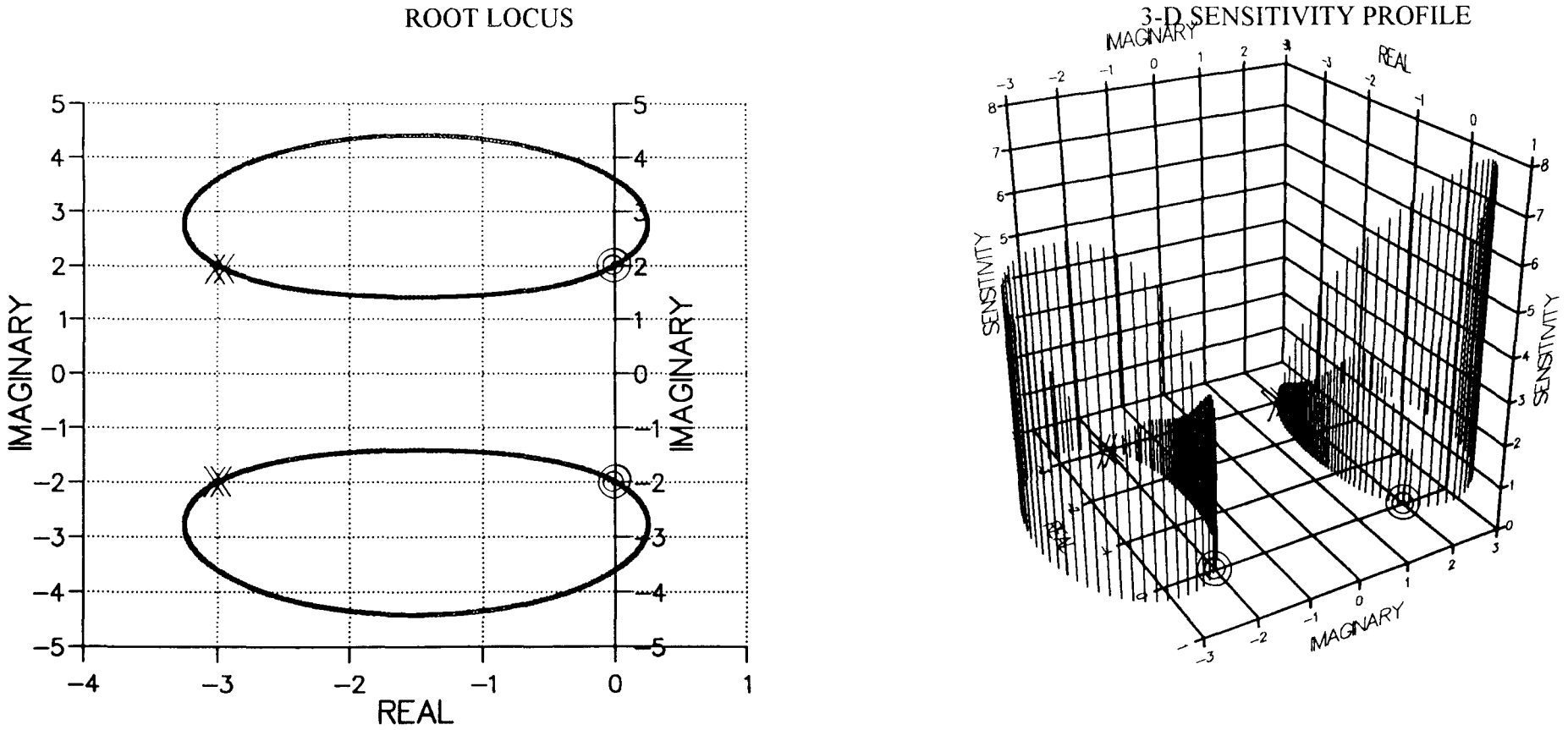


FIGURE 4.4 #D FOURTH-ORDER PLANT

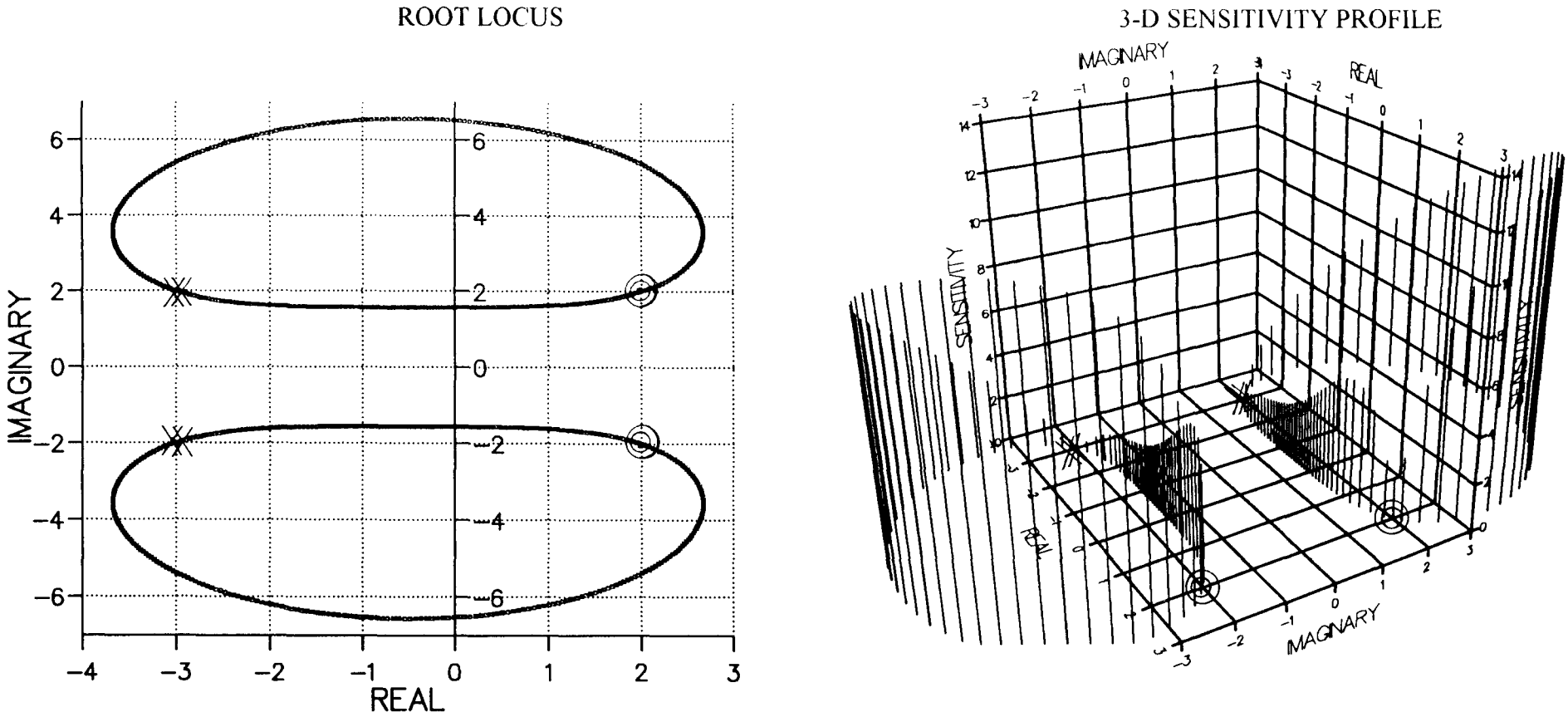
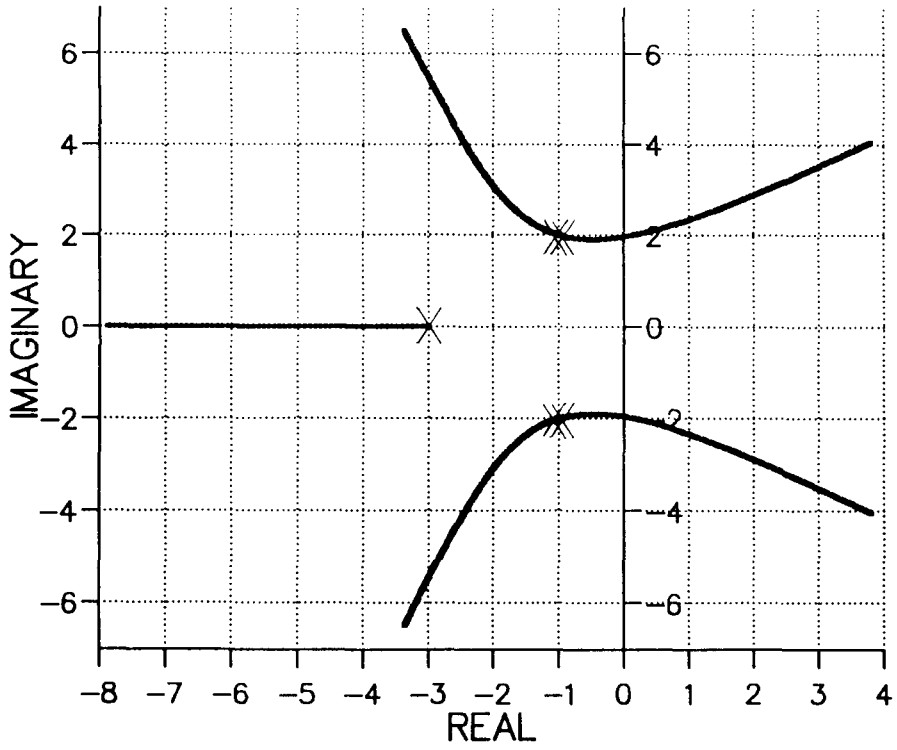


FIGURE 4.4 #E FOURTH-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

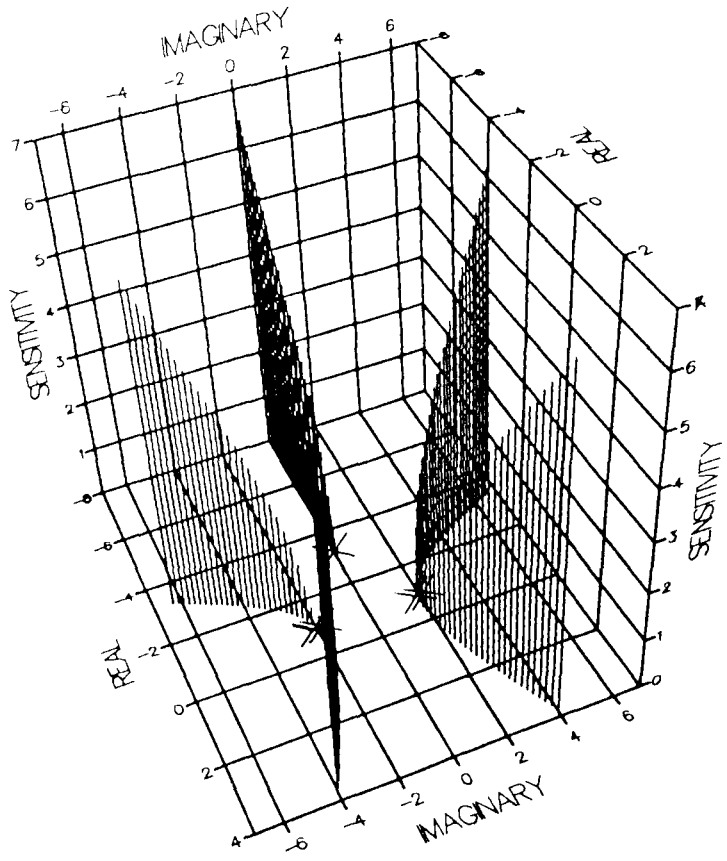


FIGURE 4.5 A FIFTH-ORDER PLANT

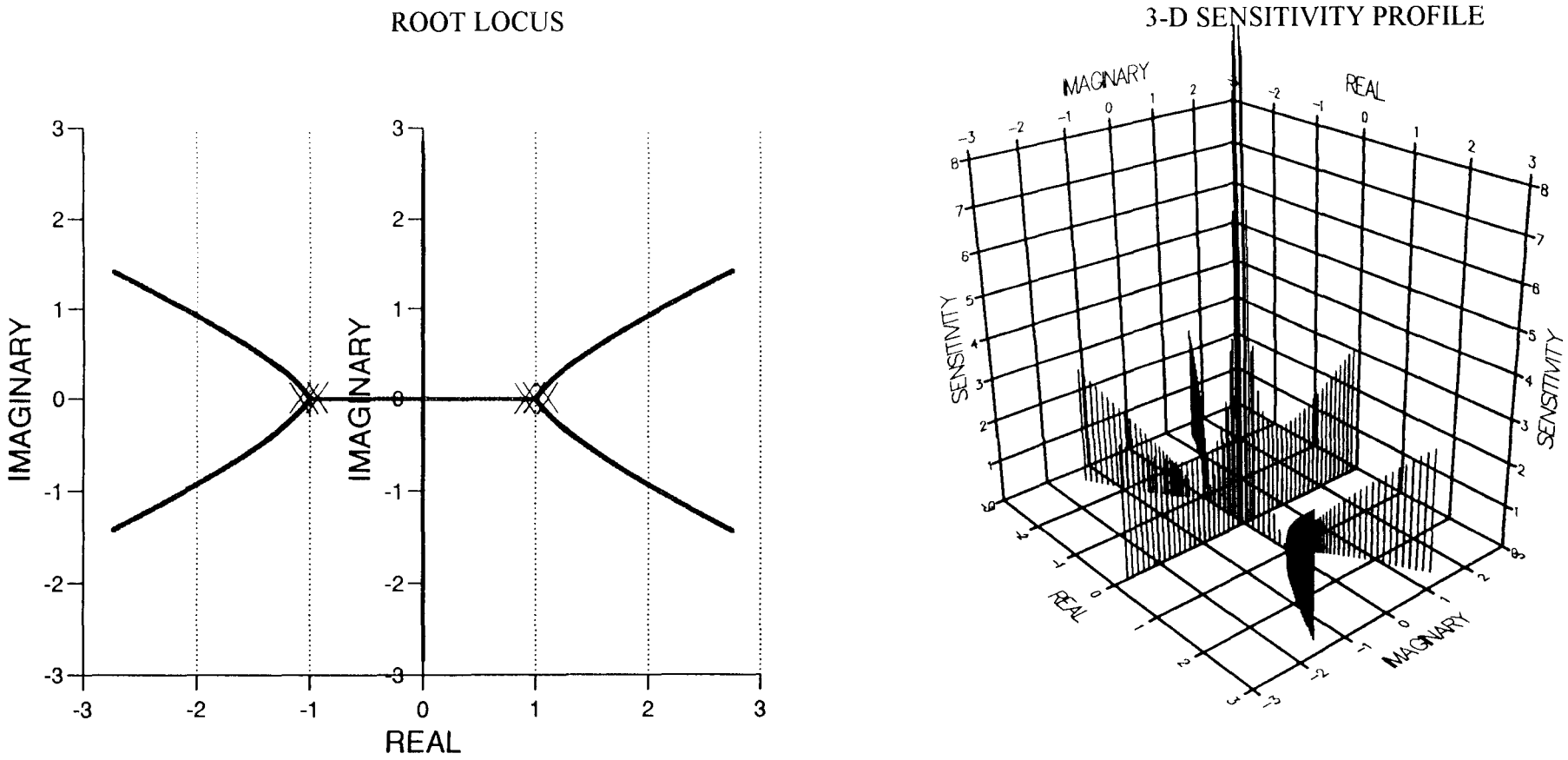
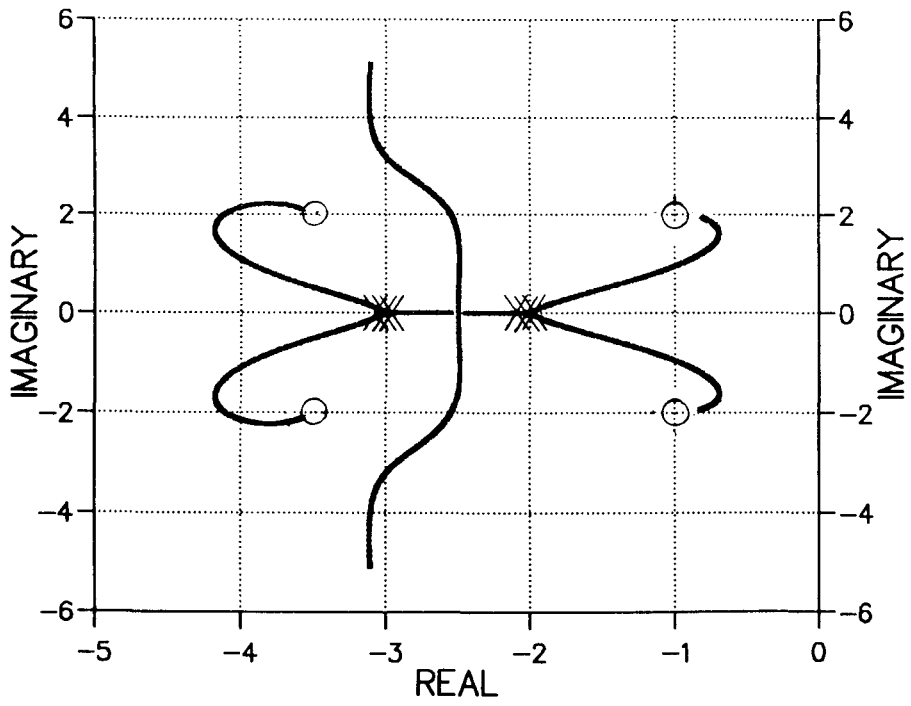


FIGURE 4.6 A SIXTH-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

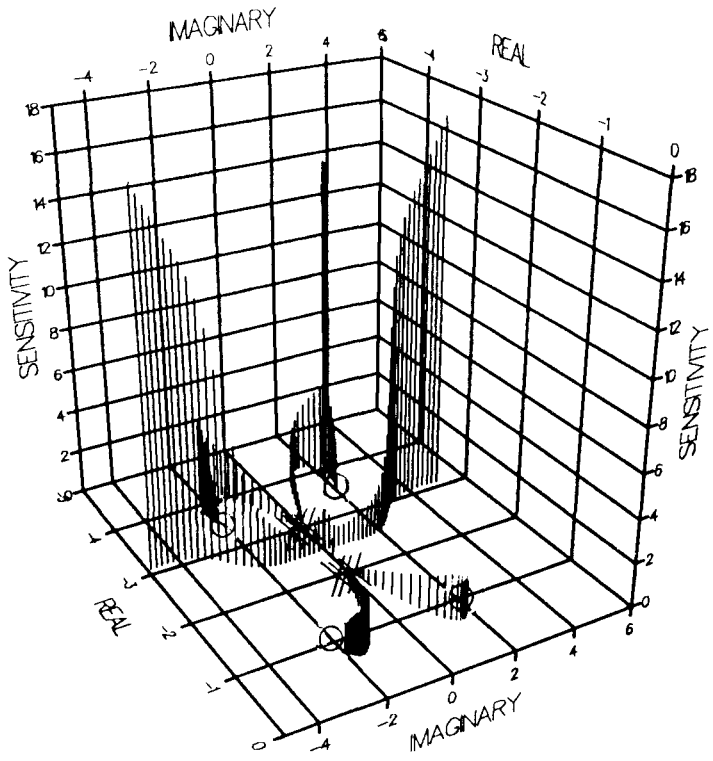


FIGURE 4.6 B SIXTH-ORDER PLANT

ROOT LOCUS

3-D SENSITIVITY PROFILE

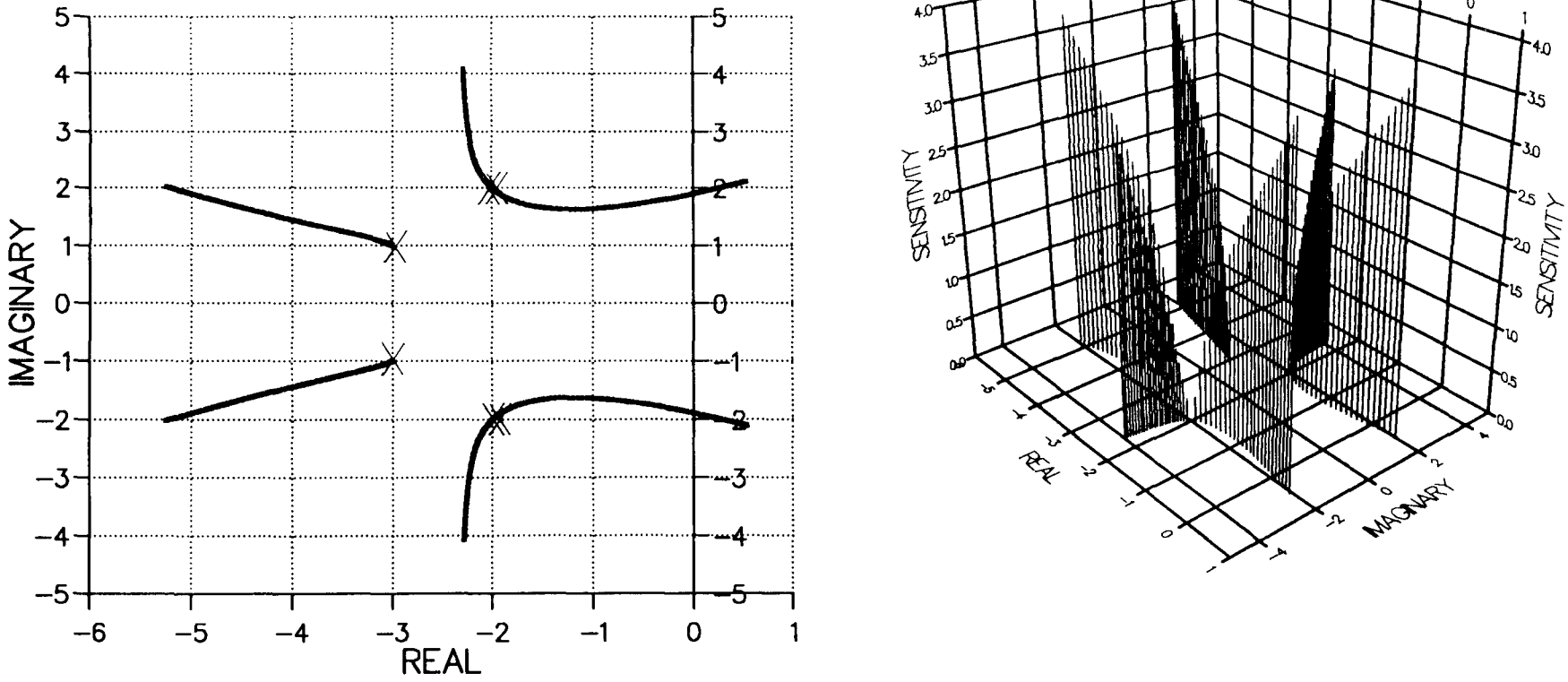


FIGURE 4.6 C SIXTH-ORDER PLANT

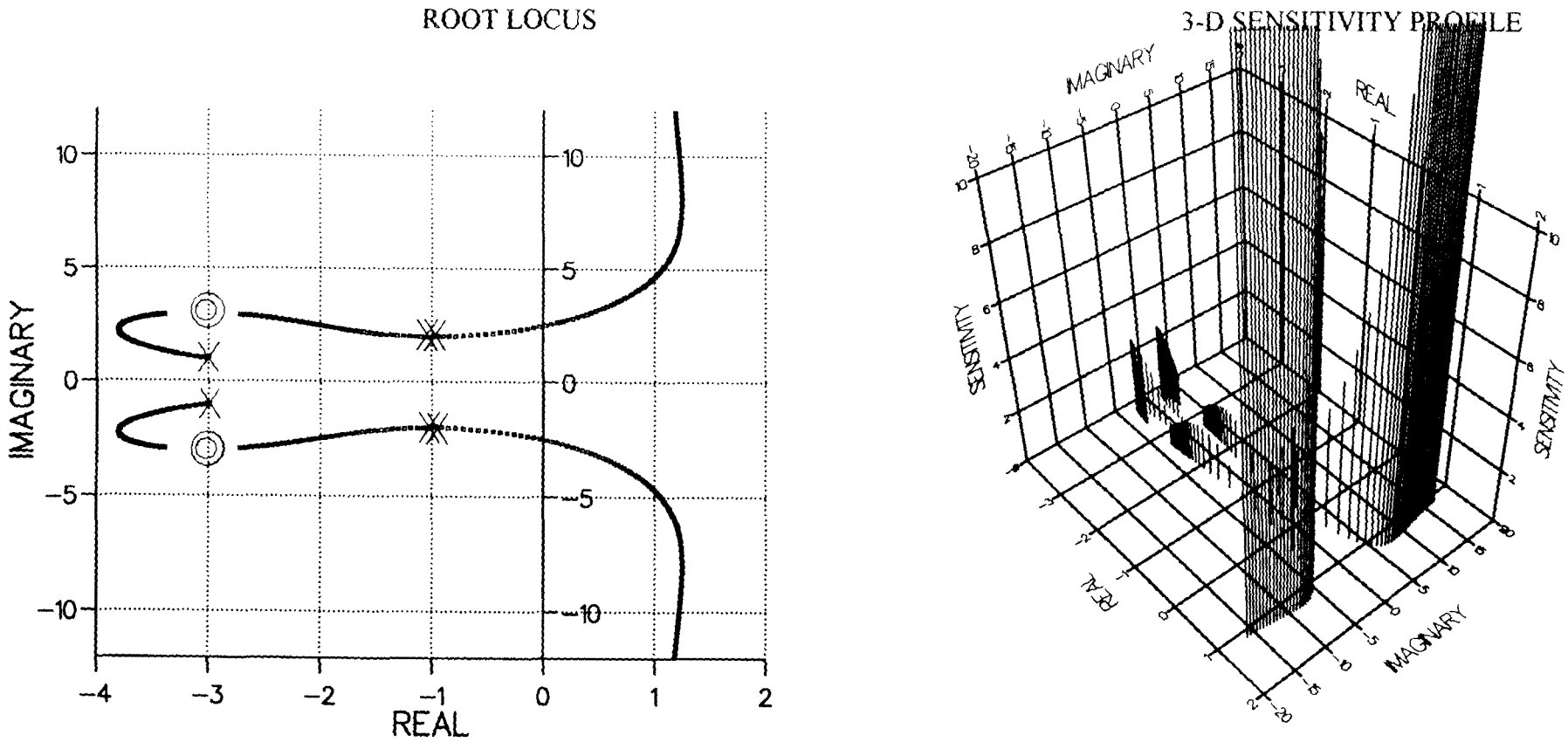


FIGURE 4.6 D SIXTH-ORDER PLANT

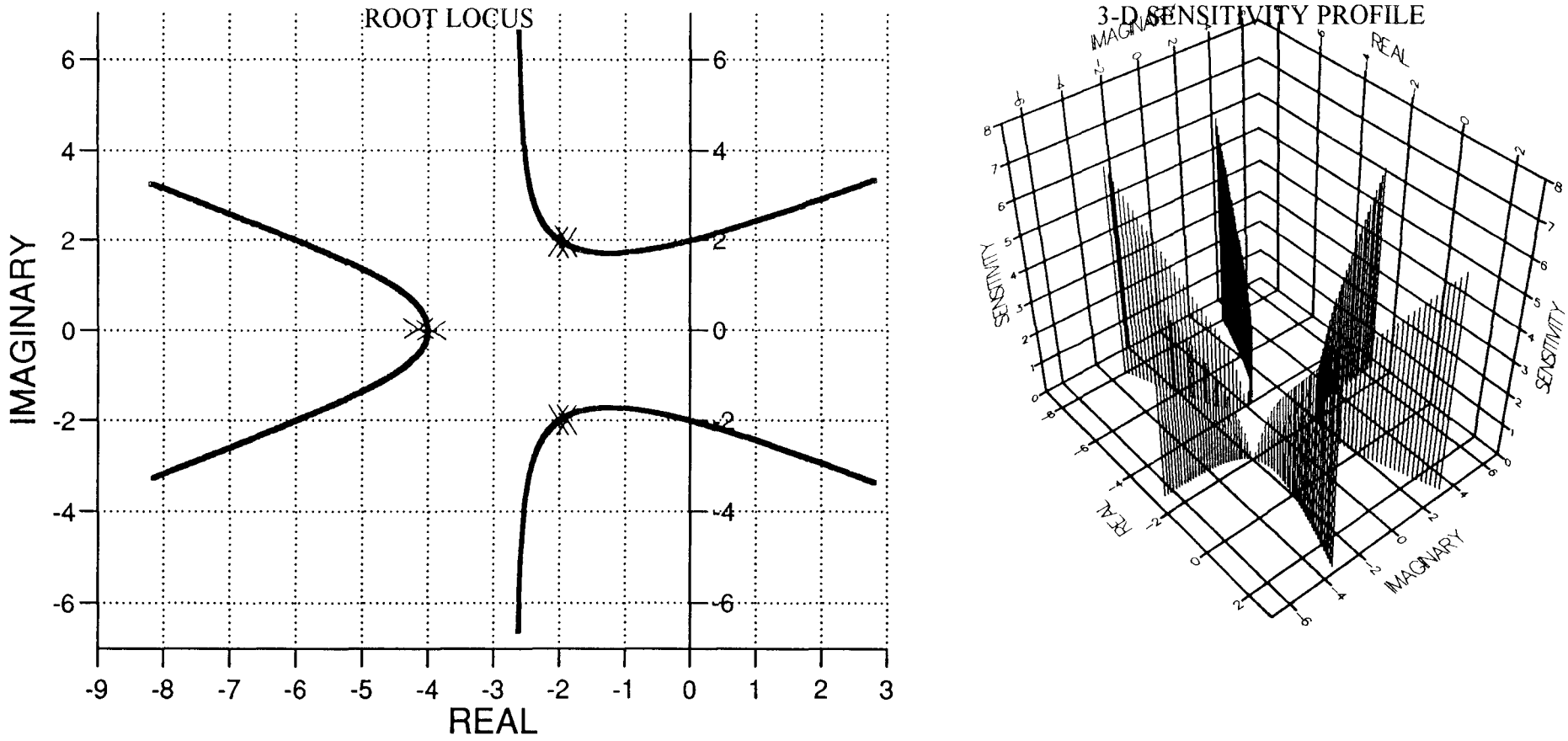


FIGURE 4.6 E SIXTH-ORDER PLANT

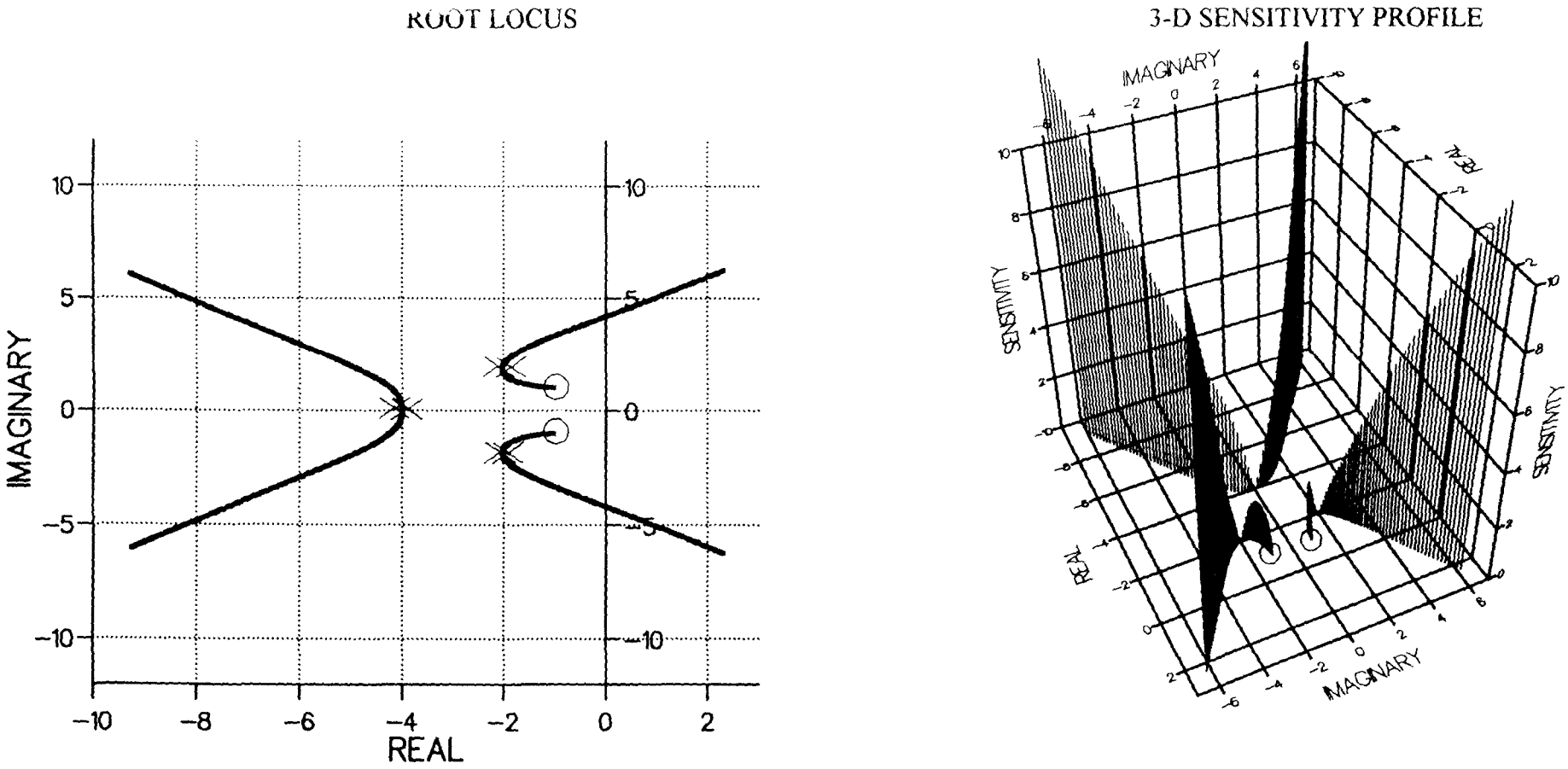
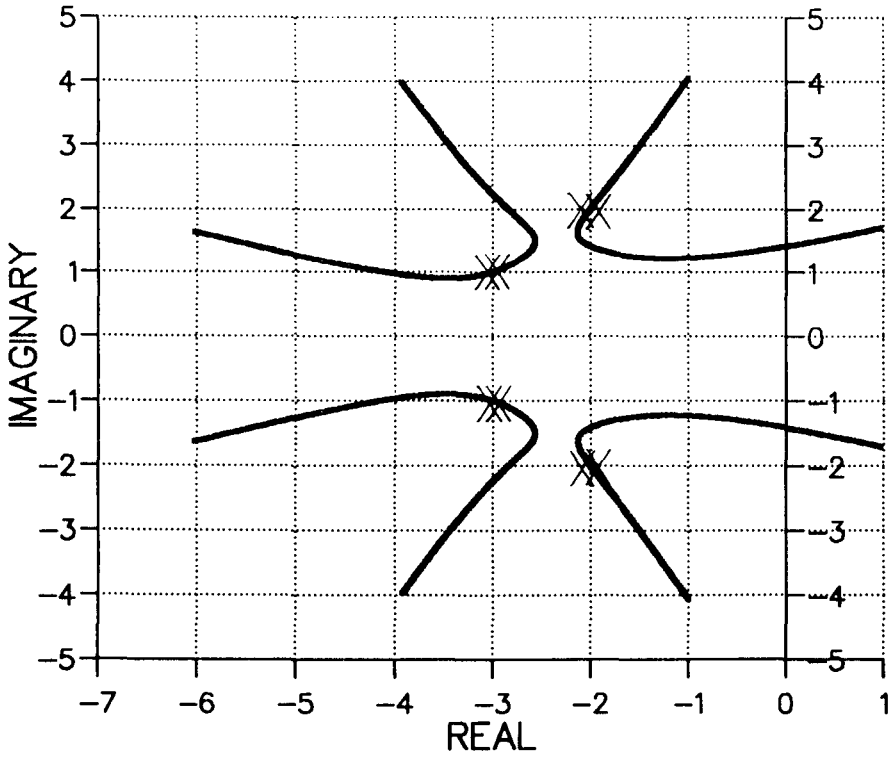


FIGURE 4.6 F SIXTH-ORDER PLANT

ROOT LOCUS



3-D SENSITIVITY PROFILE

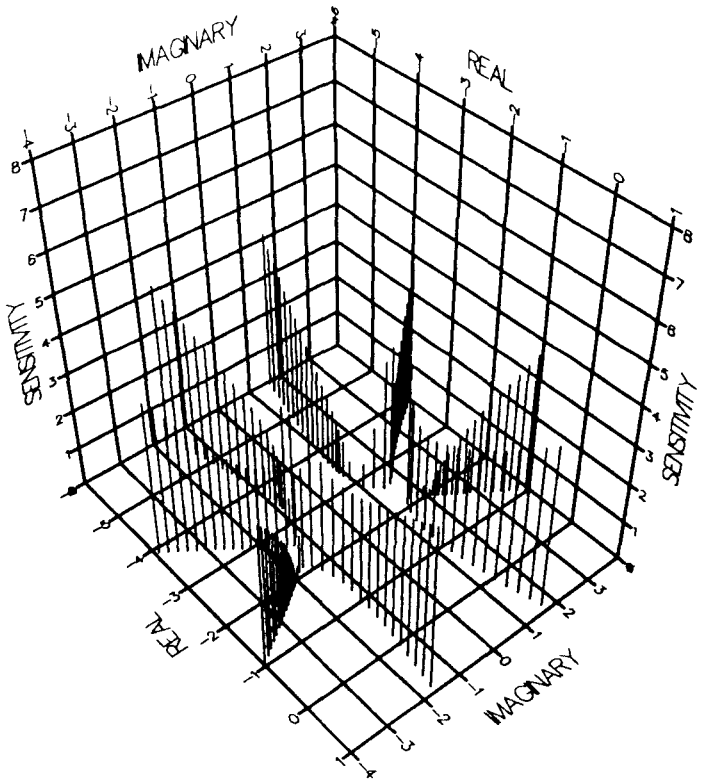


FIGURE 4.8 A EIGHTH-ORDER PLANT

Appendix C

Program Listing To Calculate The Mean-Deviation And Mean-Variance Of Two Sets Of Data

```
/*****************************************************************
* This program calculates the mean deviation and the mean        *
* variance of two sets of data; read from two data files         *
*----------------------------------------------------------      *
*    Author:            N. Golesorkhi                            *
*    Created on Date: January 1992                               *
*    Last Revised:      February 1993                            *
*****************************************************************/

#include<math.h>
#include<stdio.h>
float x1[1000],temp[1000],x2[1000],diff[1000],sqdiff[1000],
sumdiff,sumsqdiff;
int i=0,a=0,b=0;
FILE *in1,*in2;

main()
{
if( (in1=fopen("a:$$t21.y","rt"))==NULL)
   {
   fprintf(stderr, "cannot open input file.\n");
   return(1);
   }
if( (in2=fopen("a:$$t22.y","rt"))==NULL)
   {
fprintf(stderr, "cannot open input file.\n");
   return(1);
   }
while(!feof(in1))
   {
   fscanf(in1,"%f %f",&x1[i],&temp[i]);
   fscanf(in2,"%f %f",&x2[i],&temp[i]);
   i++;
   }
fclose(in1);
fclose(in2);
for(b=0,a=3;a<i;a++,b++)
   {
   diff[b]=fabs(x1[a]-(x2[a]));
   sqdiff[b]=diff[b]*diff[b];
   }
sumdiff=0;
sumsqdiff=0;
for(b=0;b<i;b++)
   {
sumdiff=sumdiff+diff[b];
sumsqdiff=sumsqdiff+sqdiff[b];
   }
```

```
sumdiff=sumdiff/(i-3);
sumsqdiff=sumsqdiff/(i-3);
printf("\n\n\n\n++++++++++++++++++++++++++++++++++++++++++++++++++");
printf("\n Sum of differences (Mean deviation)  = %f",sumdiff);
printf("\n Sum of Sq-differences (mean variance)= %f",sumsqdiff);
printf("\n\n No. of data points  %d",i-3);
printf("\n++++++++++++++++++++++++++++++++++++++++++++++++++");
printf("\n\n                Press Any Key To Exit");
getch();
}□
```

References

1. V.Gourishankar And K. Ramar "Pole Assignment With Minimum Eigenvalue Sensitivity To Plant Parameter Variations ", Int. J. Control, 1976, Vol., 23, No.4, PP. 493-504.

2. Rynaski, E., " Autom. Control Conference Workshop", University Of Michigan, 1968, PP. 193.

3. N. Golesorkhi And F.B. Khalafallah, "Sensitivity-Based Algorithm For Modelling Of Plants With Uncertainty. " IEE Colloquium On Genetic Algorithms For Control And Systems Engineering, May 1992, PP.12/1 -12/11.

4. N. Golesorkhi And F.B. Khalafallah, "Modelling Of Communication Systems With Uncertainty". Proceeding Of 13th International Congress On Cybernetics , Aug. 1992, Namur, Belgium.

5. N. Golesorkhi And F.B. Khalafallah, "Fault Simulation Of Communication Systems Using A Sensitivity Based Approach". Proceeding Of 13th International Congress On Cybernetics , Aug. 1992, Namur, Belgium.

6. B.S.Morgan, Jr., "Sensitivity Analysis And Synthesis Of Multivariable System", IEEE Transactions On Automatic Control, July 1966, Vol. Ac-11, No. 3.

7. D. C. Reddy, "Eigenfunction Sensitivity And The Parameter Variation Problem", Int. J. Control, 1969, Vol. 9, PP.561-568.

8. T. J. Owens And J. O'reilly "Parametric State-Feedback Control For Arbitrary Eigenvalue Assignment With Minimum Sensitivity", IEEE Proceedings, Nov. 1989, Vol. 136, Pt. D, No. 6.

9. S. L. Shah, D. G. Fisher And D. E. Seborg "Eigenvalue Invariance To System Parameter Variations By Eigenvector Assignment", Int. J. Control, 1977, Vol., 6, PP.871-881.

10. H. Qui And V. G. Gourishankar "Design Of Optimal Feedback Controllers For Minimum Eigenvalue Sensitivity", Optimal Control Applications & Methods. 1984, Vol., 5, PP.309-317.

11. C. Verds & A. O. Marttelo "Sensitivity Minimisation With Closed Loop Eigenvalues Assignment", IEE. Conference Control Scotland , March 1991, PP.756-759.

12. K. D. Wellstead And S. Daley "On Finding Polynomial Controllers With Reduced Pole Sensitivity Using State Space Methods", IEE. Conference On Control, March 1991, PP.677-681.

13. G. R. Duan "A Simple Algorithm For Robust Pole Assignment In Linear Output Feedback", IEE. Conference Scotland, March 1991, PP.682-686.

14. J. Kautsky, N. K. Nicholes And P. V. Doorens "Robust Pole Assignment In Linear State Feedback", Int. J. Control, Vol. 41, 1985, No. 5, PP.1129-1155.

15. J. W. Howze, And R. K. Cavin "Regular Design With Modal Insensitivity", IEEE Transaction On Automatic Control, June 1979, Vol. Ac-24. No. 3, PP.466.

16. R. K. Cavin And S. P. Bhattacharyya "Robust And Well Conditioned Eigenstructure Assignment Via Sylvester's Equation", Optimal Control Applications & Methods, 1983, Vol. 4, PP.205-212.

17. T. R. Crossley And B. Porter "Eigenvalue And Eigenvector Sensitivities In Linear Systems Theory", Int. J. Control, 1969, Vol. 10, No. 2, PP.163-170.

18. B. S. Chen, S. S. Wang And H. C. Lu "Minimal Sensitivity Perfect Model Matching Control", IEEE Transaction On Automatic Control, December 1989, Vol. 34, No. 12.

19. P. M. Gahinet, A. J. Laub, C. S. Kenney, And G. A. Hewer "Sensitivity Of The Stable Discrete-Time Lyapunov Equation", IEEE Transaction On Automatic Control, November 1990, Vol. 35, No. 11.

20. A. Test And A. Viccino "Robust Stability Of State-Space Models With Structured Uncertainties", IEEE Transaction On Automatic Control, February 1990, Vol. 35, No. 2.

21. C. S. Burger "Robust Control Of Discrete Systems", IEE Proceedings, July 1989, Vol. 136, Pt. D. No. 4.

22. H. Chapellat, M. Dahleh, And S. P. Bhattacharyya "Robust Stability Under Structured And Unstructured Perturbations", IEEE Transaction On Automatic Control, October 1990, Vol. 35, No. 10.

23. A. Gandini "Importance & Sensitivity Analysis In Assessing System Reliability", IEEE Transactions On Reliability, April 1990, Vol. 39, No. 1.

24. T. L. Landers, H. A. Taha And C. L. King " A Reliability Simulation Approach For Use In The Design Process", IEEE Transaction On Reliability , June 1991, Vol. 40, No. 2.

25. T. A. Musson "Design Rules For Reliability & Maintainability In Electronics", IEEE Transaction Reliability, December 1987, Vol. R-36, No. 5.

26. H. J. Kennedy "Reliability And Maintainability In Computer-Engineering General Chairman's Preface", Transactions On Reliability, December 1987, Vol. R. No. 5.

27. A. L. Reibman "Modelling The Effect Of Reliability On Performance", IEEE Transactions On Reliability, August 1990, Vol. 39, No. 3.

28. J. Healy "Statistic, Reliability, And Uncertainty", IEEE Transaction On Reliability December 1990, Vol. 39, No. 5.

29. Y. Levendel "Reliability Analysis Of Large Software Systems: Defect Data Modelling", IEEE Transaction On Software Engineering, February 1990, Vol. 16, No. 2.

30. P. Gottfried "Some Thought-Experiment In Reliability", IEEE Transactions On Reliability, June 1991, Vol. 40, No. 2.

31. Editorial "Sensitivity Analysis", IEEE Transaction On Reliability," June 1991, Vol. 40 No. 2.

32. F. Belli And P. Jedrzejowicz "Fault-Tolerant Programs And Their Reliability", IEEE Transactions On Reliability, June 1990, Vol. 39, No. 2.

33. A. Goyal, V. F. Nicola, A. N. Tantawi And K. S. Trivedi "Reliability Of Systems With Limited Repairs", IEEE Transaction On Reliability, June 1987, Vol. R-36 No. 2.

34. Editorial "Reliability: Whence & Whither", IEEE Transactions On Reliability, December 1989, Vol. 38, No. 5.

35. Editorial "Engineering, Modelling, And Arithmetic", IEEE Transactions On Reliability, April 1989, Vol. 38, No. 1.

36. K. W. Lee, J. J Higgins And F. A. Tillman "Stochastic Models For Mission Effectiveness", IEEE Transactions On Reliability, August 1990, Vol. 39, No. 3.

37. Y. B. Shieh, D. Ghosal, P. R. Chintamaneni And S. K. Tripathi "Modelling Of Hierarchical Distributed Systems With Fault-Tolerance", IEEE Transactions On Software Engineering, April 1990, Vol. 16, No. 4.

38. J. J. Leary, And P. J. Gawthrop "Process Fault Detection Using Constraint Suspension", IEE Proceedings, July 1987, Vol. 134, Pt. D. No. 4.

39. J. H. Kim And Z. Bien "Geometric Approach For Fault Diagnosis In Linear Dynamic Control Systems", IEE Proceedings-D. May 1991, Vol. 138. No. 3.

40. F. Belli And K. E. Groksspietsch "Specification Of Fault-Tolerant System Issues By Predicate/Transition Nets And Regular Expressions-Approach And Case Study", IEEE Transactions On Software Engineering," June 1991, Vol. 17, No 6.

41. C. Wel " Fuzzy Systems Identification", IEE Proceedings, July 1989, Vol. 136, Pt. D. No. 4.

42. A. Hauslein And B. Page "Knowledge-Based Approach To Modelling And Simulation support", Syst. Anal. Model. Simul., 1991, 8 4/5, 257-272.

43. J.R. Calvo-Ramon, "Eigenstructure Assignment By Output Feedback And Residue Analysis", IEEE Trans. Automat. Contr., 1986, Vol. Ac-31, PP.247-249.

44. W.R. Evans, "Graphical Analysis Of Control Systems", Trans. AIEE, 1948, Vol.67, PP.547-531.

45. R.A. Ash And G.R. Ash, "Numerical Computation Of Root Loci Using The Newton-Raphson Technique", IEEE Trans. Automat. Contr., 1968, Vol. ,PP.576-582.

46. L.Batista And W.D. Picchia, "A Novel Root Locus Calibrated In Real Gain In The Inverse S-Plane", IEEE Trans. Automat. Contr., 1990, Vol.35, PP.957-962.

47. D.A. Pierre,"Closed-Form Root-Loci For Quasi-Dipoles With Control Design Applications", IEEE Trans.Automat. Contr., 1984, Vol. Ac-29, PP. 824-827.

48. Z. Bien And J. Lee, "A Note On A Computer-Aided Root-Locus Method", IEEE Trans. Automat. Contr., 1986, Vol. Ac-31, PP. 246-247.

49. T Williams, "Optimal Root-Loci Of Flexible Space Structures", IEEE Trans. Automat. Contr., 1991, Vol. 36, PP. 375-377.

50. H.D. Albertson, B.F. Womack, "On Computer Implementation Of Analytic Root-Locus Plotting", IEEE Trans. Automat. Contr., 1968, PP. 744-745.

51. M.J. Maro, "Numerical Analysis: A Practical Approach", New York: Macmillan, 1982.

52. D.K. Lindner, J. Babendreir, A.M.A. Hamdan, "Measure Of Controllability And Observability And Residues", IEEE Trans. Automat. Contr., 1989, Vol. 34, PP. 648-651.

53. J.F. Aurand, "An Algorithm For Computing The Roots Of A Complex Polynomial", IEEE Trans. Automat. Contr., 1987, Vol. Ac-32, PP. 164-166.

54. J.C. Doyle, B.A. Francis And A.R. Tannenbaum, "Feedback Control Theory", Macmillan Publishing Co., 1992.

55. N.K. Sinha, "Control Systems", Hrw International, 1986.

56. G.F. Franklin And J.D. Powell, "Digital Control Of Dynamic Systems", Addison Wesley, 1980.

57. P.M. Thompson, "Program Cc Version 4 Manuals", System Technology Inc. Hawthorne Ca, 1989.

58. Borland International, "Turbo C Software, User Guide And Reference Manuals Version 2.0", 1988.

59. L. Hancock And M. Krieger, "The C Primer", Mc Graw-Hill, 1986.

60. B.W. Keringhan And D.M. Ritchie," The C Programming Language", Prentice-Hall Inc. 1978.

80. A. A. Davidov, " Singularities In Two-Dimensional Control Systems (In Russian). Moscow State University, 1982,149 P.

81. J. G. Woyka, " The Two Cultures Reconciled", Proceeding Of International Congress On Cybernetics, Namur, Belgium, 1992 (In Press).

82. J. G. Woyka, " The Two Cultures Reconciled", Study Of Randomness ,Logic Of Prediction, University Of Edinburgh, P. 251, 1974.